Geometrically and Perceptually Accurate Facial Mesh Synthesis and Personalised Blendshapes Generation using Graph Neural Networks

submitted by

ROBERT KOSK

for the degree of

Doctor of Engineering Digital Media

of

BOURNEMOUTH UNIVERSITY

Faculty of Media and Communication

Centre for Digital Entertainment

March 2024

Centre for Digital Entertainment

Bournemouth University

HUMAIN

This page is intentionally left blank.

Supervisors

Dr Richard Southern (Bournemouth University)
Prof. Lihua You (Bournemouth University)
Willem Kokke (Humain Limited)

This page is intentionally left blank.

ABSTRACT

The importance of geometric deep learning applications to 3D content creation has increased rapidly, driven by significant investments in the next generation Virtual Reality platforms and Visual Effects intensive productions. Generation of high fidelity digital humans became a focal point and one of the fundamental challenges in these applications. The availability of high-quality facial scans and recent advances in deep learning methods applied to mesh processing have led to the development of data driven models. These approaches are at the forefront of content creation technologies, offering artists and users the ability to rapidly generate and edit character assets. Despite major improvements in recent years, the geometric and perceptual quality of 3D facial meshes generated with these techniques do not meet high standards of VFX and AAA video games industry.

This research explores geometric deep learning approaches to generation and editing of registered 3D facial meshes and personalised blendshapes. Significant impact of facial shape representations on the quality of reconstructed meshes is demonstrated. Novel methods are proposed to improve the geometric and perceptual accuracy of generated facial meshes and personalised blendshapes. Additionally, user parameters are exposed to independently edit low and high frequency facial deformations.

The concept of Deep Spectral Meshes is introduced, which is based on spectral decomposition of meshes in 3D shape representation learning. Using the proposed framework, a parametric model for 3D facial mesh synthesis is built to demonstrate improvements in facial mesh reconstruction in terms of geometric and perceptual error metrics. Additionally, a method is proposed to leverage mutually exclusive objectives of independent control of deformations at different frequencies, and generation of plausible, synthetic examples.

A platform is built to compare various deep 3D Morphable Models coupled with different 3D mesh representations and to evaluate them with several distance and perceptual metrics. Using the platform, improvements upon existing state-of-the-art reconstruction results are demonstrated and strengths and weaknesses of 3D mesh representations and preprocessing techniques are further exposed. Subsequently, personalised blendshapes generation with spectral mesh processing method is introduced to improve geometric

and perceptual accuracy of synthesised expressions. The proposed method improves personalisation over the deformation transfer, which remains a standard industrial practise.

The projects presented in this thesis address professional requirements of industrial partner, Humain Ltd.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

This page is intentionally left blank.

ACKNOWLEDGMENTS

I have to start by thanking my supervisor **Richard Southern** for many years of truly transformative mentorship, shaping the way I think and solve problems. I would like to extend my deepest gratitude to my supervisor **Lihua You** for invaluable advice and dedication.

I am extremely grateful to **Greg Maguire** for constant inspiration and guidance in making this work impactful in industrial setting. Thank you for all the resources and the opportunity of working in such a research and innovation-driven environment. I would like to thank my industrial supervisor **Willem Kokke**, whose technical wizardry has helped resolve the toughest implementation issues.

Special thanks to my colleagues from Humain who I worked with and learned from. Particularly, I want to thank **Shaojun Bian** and **Anzong Zheng** from Humain's R&D team, for countless discussions about my work and showing me how to be a successful researcher.

I would like to extend my sincere thanks to everyone at the Centre for Digital Entertainment, especially **Mike Board** for all-important support and encouragement, and **Zoe Leonard** for unforgettable memories and sense of belonging.

I am grateful to the reviewers of this thesis, **Xiaosong Yang** and **Vinay Namboodiri**, for their valuable feedback, which has helped improve this work.

This research would not have been possible without funding by **EPSRC** grant EP/016 540/1 and **Humain Ltd**.

Thank you to all the friends who have been close to me for all these years, particularly **Brygida**, **Șerban** and **Phoebe**.

I am also thankful to my family. I could not have undertaken this research without many sacrifices and unconditional love of my parents **Beata** and **Krzysztof**. Thank you for always believing in me and being anchors in my life. I extend my appreciation to my brother **Darek** for being my great friend and supporter.

Finally, I would like to thank my partner **Hubert** for sharing all my ups and downs over all the years of this journey. Your love and support gave me strength to finish this thesis.

This page is intentionally left blank.

DECLARATION

This thesis had been created by myself and has not been submitted in any previous application for any degree. The work in this thesis has been undertaken by myself except where otherwise stated.

Robert Kosk
March 2024

This page is intentionally left blank.

PUBLICATIONS

The following publications stem from research presented in this thesis:

Kosk, R., Southern, R., You, L. Bian, S., Kokke, W., Maguire, G. (2024). **Deep Spectral meshes: Multi-Frequency Facial Mesh Processing with Graph Neural Networks.** *Electronics* 13, no. 4: 720. https://doi.org/10.3390/electronics13040720 [Journal Article]

Kosk, R., Southern, R., You, L. Bian, S., Kokke, W., Maguire, G. (2024). **Mesh Representation Matters: Investigating the Influence of Different Mesh Features on Perceptual and Spatial Fidelity of Deep 3D Morphable Models.** *Virtual Reality & Intelligent Hardware* 6, no. 5: 383-395. https://doi.org/10.1016/j.vrih.2024.08.006 [Journal Article]

Kosk, R., Southern, R., Kokke, W. (2020). **Parametric Face Model with Geometric Deep Learning.** *The 17th ACM SIGGRAPH European Conference on Visual Media Production*, 7–8 December 2020, Virtual Conference. [Poster]

This page is intentionally left blank.

CHAPTER 1

INTRODUCTION

1.1 Motivation

Generating high-fidelity digital characters has become more significant now than ever before. With substantial investments pouring into new visual effects-intensive content by technology companies, the demand for lifelike digital humans is escalating. For example, conglomerates like *Meta Platforms, Inc.* are reported to have invested at least USD 36 billion in developing content and tools to populate the next-generation social media platforms based on Virtual Reality.

Meanwhile, video game developers express concerns over lack of highly skilled human resources required to generate high-fidelity game characters. Modelling and animation of high-quality, digital faces remains a tedious process, which requires multidisciplinary expertise and expensive, complex setups. As an example, it takes approximately 180 person-hours for an experienced 3D modeller to create 89 blendshapes used for a digital double. High-end rigs often require hundreds of shapes, scaling up to even 900 person-hours of digital sculpting, assuming that no automated process is involved. Current industry methods use deformation transfer techniques to aid the blendshapes generation process. Nonetheless, the resulting shapes require extensive and time-consuming manual clean-up by skilled artists. Moreover, expressions produced with deformation transfer techniques are not personalised to the target identity model.

Therefore, visual effects (VFX), games and animation studios look for solutions which would cut costs and time of facial shapes creation, while maintaining standards appropriate for industrial applications. At the same time, the rapidly developing virtual reality industry demands fully automated and flexible methods to generate likeness and dynamics of a human face. Digital faces are used beyond entertainment domain, with applications in medical visualisation and training, forensic facial reconstruction, as well as virtual avatars for chatbots and telepresence.

The availability of high-quality facial scans and recent advances in deep learning methods applied to mesh processing have led to the development of data-driven parametric models. These approaches are at the forefront of content creation technologies, allowing artists and users to rapidly generate high-fidelity character assets for these applications. However, geometric and perceptual quality of generated assets often does not meet standards appropriate for industrial applications.

1.2 Industrial Context

The industrial partner of this research is Humain Ltd., the supplier of 3D digital character services to world leading entertainment and technology companies within the video games industry. Founded in 2017 and headquartered in Belfast, the company collaborated on some of the biggest video game titles, including several owned by Xbox Game Studios: ZeniMax Media Inc., Obsidian Entertainment, Inc. (The Outer Worlds), The Initiative (Perfect Dark series), and 343 Industries (Halo series). Humain's credits include Call of Duty: Black Ops Cold War (Activision Publishing, Inc.), Saints Row series (Deep Silver), the critically acclaimed Diablo II Resurrected (Blizzard Entertainment), Warhammer 40,000: Space Marine 2 (Saber, Focus Entertainment), Microsoft Flight Simulator 2024 (Asobo, Xbox Game Studios) and Avowed (Obsidian Entertainment, Xbox Game Studios). Humain has also completed projects for Google Stadia and Google Alphabet and has moved into creating digital twins of music celebrities including David Guetta, Squeezie, Martin Garrix and Ne-Yo for metaverse music company, Stage11.

Currently, Humain's expertise is primarily focused on digital faces for AAA video games. Humain Ltd. does not develop its own video games. Typically, the company produces a facial rig from a 3D model of a human face provided by a game developer. Over the years, Humain Ltd. has developed EKER™ - the facial rigging system based on Facial Action Coding System [35] using scans of Dr Erika Rosenberg as a deformation source (see Figure 1.1). EKER™ takes a neutral face mesh as an input and automatically

Figure 1.1 High-resolution scan of Dr Erika Rosenberg performing the combination of AU6, AU12, AU15 and AU17. Image courtesy of Humain Ltd.

generates a facial rig. Due to automation, it reduces human intervention in the rigging process. Nonetheless, the resulting shapes are not personalised and require additional manual cleanup by the artists.

Additionally, Humain would like to be able to generate facial shapes of new identities to populate virtual worlds. The resulting faces should be editable at coarse and fine level of detail with a small set of parameters. The parametric model could be used in a 3D face reconstruction task. The research and development of a monocular 3D face reconstruction system is an ongoing process at Humain, and work presented in this thesis has been carried out in parallel, as complementary part of this process.

Humain's goals include:

1. Generation of facial shapes of high perceptual and geometric accuracy.
2. Generation of personalised blendshapes given a neutral face mesh.
3. Editing of facial shapes at coarse and fine level of detail.

1.3 Academic Context

In response to needs of the industrial partner Humain Ltd., as well as wider computer animation, VFX and games industry, the following research question has been posed:

> How to generate and edit geometrically and perceptually accurate digital faces and personalised blendshapes?

Here, **generation** of faces means an automated process of synthesising triangle meshes of head and neck.

Editing refers to modifying facial shapes using a limited number of controllers. In this work, the focus is on the ability to separately edit low- and high-frequency facial shape displacements.

Geometric accuracy is measured with point-wise distance metrics, such as L_1 and L_2 norms. These metrics are sensitive to coarse, low-frequency discrepancies between meshes.

Perceptual accuracy is measured with perceptual metrics, such as Fast Mesh Perceptual Distance (FMPD) [132] or the Dihedral Angle Mesh Error (DAME) [127]. These metrics are sensitive to fine, high-frequency discrepancies between meshes.

Digital faces are either virtual doubles of existing humans, or non-existing heads. Following the requirements of the industrial partner Huamin Ltd., as well as high-end AAA games production standards, the focus is on facial meshes with 10-20K vertices.

Personalised blendshapes are a set of facial expressions of a given identity. Observable changes in facial shape result from contraction and relaxation of underlying muscles, thickness of fascia, skin elasticity and shape of facial features, jaw and skull. In contrast, generic blendshapes do not account for these personal features.

In this thesis, it is hypothesised that challenges posed in the research question can be addressed with geometric deep learning of suitable facial shape representations.

Three-dimensional meshes are non-Euclidean data, unlike Euclidean data such as voxels, which have an underlying grid structure and can be treated by extending already-existing 2D deep learning paradigms. The lack of grid structure poses a challenge when attempting to apply classical deep learning techniques to non-Euclidean data. To address this problem, geometric deep learning [15] has been developed explicitly for non-Euclidean data. Geometric deep learning is used in this work to learn the parametric space of facial meshes. Learned parameters can be used in facial shape generation and editing.

Parametric models, such as 3D morphable models [34], are commonly used to synthesise new meshes by altering the coefficients in a parametric space. They are widely applied due to their ability to model intrinsic properties of 3D faces. Parametric models have been used in graph neural networks to represent facial shapes. Parametric models with graph neural networks, called deep 3D morphable models, are also used in this work for 3D facial mesh synthesis.

It is hypothesised that using different input and output representations to deep 3D morphable models can improve upon existing mesh reconstruction methods in terms of either perceptual or geometric accuracy. It can be expected that using input and output representations, which explicitly encode the surface properties, improves the perceptual quality of the resulting meshes. On the other hand, using input and output representations, which explicitly encode the vertex positions in 3D space, might improve the geometric accuracy of generated meshes. To prove these hypotheses, a selection of deep 3D morphable models combined with different mesh representations is compared in this thesis.

Lower-frequency displacements encode most of the mesh volume, while higher-frequency displacements describe fine surface details. This observation can be used to improve the geometric and perceptual quality of generated meshes. Spectral mesh processing [115] derives eigenvalues, eigenvectors, or eigenspace projections from the mesh operators and uses them to carry out desired tasks. It provides a powerful means to achieve different approximations of a 3D mesh with different frequencies.

It is hypothesised that by integrating the shape representation which explicitly encodes the surface properties, and by integrating spectral mesh processing for decomposition of

mesh displacements, geometric deep learning, and parametric models with graph neural networks, a new 3D facial mesh synthesis model can be developed, such that it would expose user parameters to control disentangled low- and high-frequency displacements, generate plausible facial shapes, and allow the user to control displacements independently at low- and high-frequency levels.

1.4 Objectives

Based on the aims covered in Section 1.3, the following objectives are devised:

1. Evaluate influence of different mesh representations input to graph autoencoders in terms of geometric accuracy and perceptual quality. Compare the impact of normalisation and standardisation of inputs to graph autoencoders in terms of geometric accuracy and perceptual quality. Design and implement software which allows to evaluate different inputs to various graph autoencoders.

2. Based on conclusions from Objective 1, propose and evaluate a method which utilises strengths and weaknesses of different mesh representations in graph autonecoders to improve overall perceptual and geometric accuracy of generated meshes. Demonstrate reconstruction, interpolation and editing applications of the method.

3. Using conclusions from Objective 1 and techniques developed in Objective 2, propose and evaluate a method which improves perceptual and geometric accuracy of personalised blendshapes synthesised from a neutral face input.

4. Identify potential applications and future research directions.

1.5 Contributions

Research presented in this thesis resulted in the following main contributions:

- A novel parametric deep face model which enables independent control of high- and low-frequency displacements.

- Enhanced geometric and perceptual quality of generated meshes, achieved through the use of different representations of displacements at high and low frequencies.

- Identification of strengths and weaknesses of different inputs to various graph autoencoders in terms of geometric accuracy and perceptual quality. The design and implementation of software which allows to evaluate different inputs to various graph autoencoders.

- The improved quality of generated meshes by selecting best performing combinations of an input representation and the graph autoencoder based on either perceptual or geometric quality objective.
- Introduction of spectral decomposition of meshes in 3D shape representation learning and demonstration of importance of mass matrix normalisation in this method.
- An improvement of geometric accuracy and perceptual quality of personalised blendshapes synthesis by using graph autoencoders coupled with spectral mesh processing.

1.6 Thesis Outline

The remaining chapters of this thesis are organised as follows:

Chapter 2 covers background and related work in the areas of facial shape representations, generative deep learning, deep learning on non-Euclidean domains, parametric face models and monocular 3D face reconstruction.

Chapter 3 addresses Objective 1. It investigates the hypothesis that leveraging different input and output shape representations within deep 3D morphable models can enhance mesh reconstruction methods in terms of perceptual and geometric accuracy. An evaluation platform, the Deep3DMM Comparison Platform, is introduced to assess various deep 3D morphable models across different shape representations. This chapter explores configurations of five deep 3D morphable models with four input and output representations trained on multiple datasets and evaluates them using geometric and perceptual metrics. The conclusions from this chapter provide insight into influence of different shape representations and preprocessing techniques on perceptual and geometric fidelity of reconstructed meshes.

Chapter 4 addresses Objective 2. It introduces Deep Spectral Meshes method and tests the hypothesis that by integrating the shape representation which explicitly encodes the surface properties, and by integrating spectral mesh processing for decomposition of mesh displacements, geometric deep learning, and parametric models with graph neural networks, a new 3D facial mesh synthesis model can be developed, such that it exposes user parameters to control disentangled low- and high-frequency displacements, generate perceptually and geometrically accurate facial shapes, and allows the user to control displacements independently at low- and high-frequency levels.

Chapter 5 addresses Objective 3. It examines the hypothesis that graph autoencoders can enhance the mapping between neutral face meshes and personalised expressions, outperforming existing Deformation Transfer method in terms of geometric and perceptual

accuracy. Additionally, personalised blendshapes generation with spectral mesh processing method is introduced to improve geometric and perceptual accuracy of synthesised expressions.

Chapter 6 provides a summary and conclusions. Lastly, it addresses Objective 4 through identification of potential applications and future work directions.

Finally, **Appendix A** covers software design and implementation details of the comparison platform from Chapter 3.

This page is intentionally left blank.

CHAPTER2

BACKGROUND AND RELATED WORK

In this chapter, Section 2.1 reviews important concepts and the literature on various representations of facial shapes, the deep learning methods which devise these representations in Euclidean domain are reviewed in Section 2.2 and non-Euclidean domain in Section 2.3. Parametric face models are covered in Section 2.4, and methods of mapping between facial representations, including the reconstruction of 3D face shapes from 2D images, are discussed in Section 2.5.

2.1 Facial Shape Representations

2.1.1 Images and depth

RGB and RGB-D images are the most common representations of facial shapes at a stage of acquisition. High-end facial performance capture requires expensive, multi-view camera setups. However, as monocular RGB capture devices are inexpensive and widely available, monocular methods of acquisition became an important and popular research topic. Images of faces are primarily used to infer skin texture and other facial shape representations, such as a point cloud or parameters of a parametric model. Numerous methods further constrain inference of facial shapes from images with additional information, such as landmarks, edges and silhouettes. Comprehensive review of facial landmarks detection from 2D images is provided in survey by Wu and Ji [138]. Browatzki and Wallraven [16] achieve state of the art results in predicting facial landmark heatmaps from 2D image representation of faces. Firstly, they capture implicit knowledge of facial shape by unsupervised training of an adversarial autoencoder. Then, they retask the generation of RGB facial images to prediction of landmark heatmaps. Applications of the constraints devised from 2D images are described in Section 2.5. Thanks to structured representation of 2D images, large number of deep learning models can directly operate on them.

2.1.2 Points and meshes

Point clouds, here denoted with $\mathbf{P}$, can be extracted from a set of images through Simultaneous Localisation and Mapping (SLAM) [37] [36]. Using this approach requires subjects to maintain a static facial expression. When captured in high resolution, results from SLAM often act as the ground truth of a 3D facial shape. Nonetheless, points do not provide direct information about surfaces which they represent and can be ambiguous. To give an insight into topological properties of the object, a 3D mesh representation is used. In this thesis,triangular meshes are considered and denoted as sets $\mathscr{M} = \{\mathscr{V}, \mathscr{E}\}$, where $v_i \in \mathscr{V}, i = 1 \dots |\mathscr{V}|$ is a set of vertices (points in 3D space) and $\mathscr{E}$ is a set of pairs of vertices, which define mesh connectivity. Positional information about vertices is represented using matrix $\mathbf{P}$, where rows $\mathbf{p}_i$ correspond to vertices, and columns correspond to xyz coordinates. In Section 2.3 on geometric deep learning, words *mesh* and *graph* are used interchangeably, both denoted with $\mathscr{M}$. When meshes are used as a shape representation in deep learning, some of the negative effects of their variance to rigid transformations can be reduced with rigid registration of all the training examples.

Blendshapes

3D facial shapes are usually animated in two common representations: skinning, blendshapes or their combination. Blendshapes are 3D facial meshes sharing the same connectivity, each with a different elementary expression, often derived from FACS. Animation with blendshapes is possible through blending these expressions with each other and with a base mesh. Given neutral identity face $\mathbf{B}_0$ and blendshapes $\mathbf{B}_i$, where $i \in \{1, 2..., n\}$, a new expression $\mathbf{p}$ is the linear combination of blendshapes:

$$\mathbf{P} = \sum_{i=0}^{n} \alpha_i \mathbf{B}_i \quad , \tag{2.1}$$

where α_i is a weight of a blendshape $\mathbf{B}_i$, where $0 < \alpha_i < 1$, $\forall i$ and $\sum_i \alpha_i = 1$. To be able to apply the same expressions on top of different identities, delta blendshapes are used:

$$\mathbf{P} = \mathbf{B}_0 + \sum_{i=0}^{n} \alpha_i (\mathbf{B}_i - \mathbf{B}_0) \quad . \tag{2.2}$$

This way, a model of facial shapes is obtained, where each shape can be defined in terms of non-orthogonal vectors. These vectors are useful for artists, as they are semantically meaningful. Alternative formulations with orthogonal basis are described in Section 2.4.1.

Despite its popularity, mostly attributed to simple formulation, facial animation with blendshapes has numerous limitations. Linear blending between shapes does not conform with biomechanical motion of faces. Furthermore, editing of such animations requires

the manipulation of individual blendhapes. As blendshapes do not form an orthogonal space, their linear combinations are often counter-intuitive and most combinations require additional, corrective blendshapes. In high-end facial rigs, the number of corrective blendshapes can reach over a thousand shapes, making these rigs very challenging to maintain and develop.

Facial Action Coding System

Figure 2.1 Left: muscles underlying the AUs 1-7. Right: muscular actions which change the appearance of a face. Note that FACS measures changes in facial appearance, and not muscle activations. Reproduced from the FACS manual by Ekman et al. [35].

Facial Action Coding System (FACS) is the standard taxonomy of facial expressions. It codifies movements of underlying muscles based on changes in facial appearance, as shown in Figure 2.1. Almost every facial expression can be encoded with a combination of Action Units (AUs), which are observable effects of contraction or relaxation of a certain muscle or a group of muscles. For example, a combination of AU6, AU12, AU15 and AU17 results in the expression depicted in Figure 1.1. Furthermore, each AU has its intensity score ranging from A (trace) through C (marked pronounced) to E (maximum). Importantly, FACS was not designed to codify visemes, so that additional primary shapes are required to describe speech.

Deformation Transfer

Modelling of hundreds of blendshapes, including corrective shapes, is laborious and often infeasible task. Delta blendshapes, described before, can be reused in limited extent, usually on neutral faces with very similar physiognomy and style. Seminal work of Sumner and Popovic [120] introduced the Deformation Transfer, which transfers deformations from a source subject onto a target subject. Both, source and target, do not need to share the same connectivity or number of vertices. In practice, given one set of blendshapes, artists

can automatically generate a corresponding set of blendshapes for another individual. The method generalises to thoroughly different identity shapes, including highly stylised characters. Nevertheless, selecting a set of vertex markers to establish correspondence between shapes remains a manual part of this process. Furthermore, the deformation transfer on faces ignores the fact that different individuals perform the same facial expressions in a different way. This problem is recognised in this thesis and following Objective 3, the correspondence between identity and expression is established in Chapter 5. Li et al. [77] propose to re-introduce individuality of facial expressions with a set of example poses of a target face. Yet, their technique requires additional modelling effort and improvements are seen mostly on expressions covered by example expressions, with minor impact on remaining shapes.

2.1.3 Differential surface representation

Previously described representations have one common disadvantage - their editing can be tedious due to large number of parameters. Laplacian coordinates [115] allow for new mesh editing and shape approximation techniques. As this representation inspired further work in representing displacements and some of its concepts are shared with graph convolutional networks, it deserves a more detailed overview.

There are numerous representations which allow for mesh editing, for instance free-form deformation (FFD) embeddings, finite element method (FEM) etc. Nonetheless, differential coordinates are especially interesting for our problem, as intrinsic surface properties, such as curvature, are natively optimised in this representation. The differential coordinates δ_i of the vertex $v_i \in \mathcal{V}$ with position $\mathbf{p}_i$ are calculated as follows:

$$\delta_i = w_{ij} \sum_{j \in \mathcal{N}_i} \mathbf{p}_i - \mathbf{p}_j \ , \tag{2.3}$$

where $\mathcal{N}_i$ are indices of vertices in neighbourhood of v_i. Simply put, δ_i is the difference between the vertex position and weighted sum of its neighbourhood positions. Sorkine [115] uses valence-based weights $w_{ij} = \frac{1}{|\mathcal{N}_i|}$, however cotangent weights are often used instead [84]. Differential coordinate δ_i encodes important local properties: it approximates the normal direction of a local surface and its magnitude is proportional to local mean curvature. While Laplacian coordinates are translation invariant, they are still sensitive to rotations.

Graph Laplacian

The Laplacian operator $\mathbf{L}$ transforming vectors in absolute Euclidean coordinates to differential coordinates is

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A} \ , \tag{2.4}$$

where $\mathbf{A}$ is an adjacency matrix of a mesh $\mathscr{M}$, $\mathbf{D}^{-1}$ is an inverse degree matrix of $\mathbf{A}$ and $\mathbf{I}$ is an identity matrix. As differential representation is translation invariant, the reconstruction from Laplacian coordinates back to absolute Euclidean coordinates does not have a unique solution and at least one positional constraint must be added to the linear system.

Least-squares Meshes

Sorkine and Cohen-Or [117] introduced least-squares meshes - a smooth approximation of control points, which constrain the mesh in differential representation. Positional constraints used for reconstruction are either a selection of existing points or new vertex positions, known as anchors. The system reconstructing the points is then solved in a least-squared sense. When representing facial performance, curvature is an important property to preserve. Therefore, a set of anchors can be used to encode different facial expressions and identities as least-squares meshes.

2.1.4 Deformation representation

Mesh difference is a natural way of describing facial action units, which are displacements of a neutral identity. While differential coordinates encode local surface properties, deformation representation encodes local deformation properties using deformation gradient.

Let's consider a vertex $v_i \in \mathscr{V}$ with position $\mathbf{p}_i$ on a reference mesh $\mathscr{M}_0$ and the same vertex with position $\mathbf{p'}_i$ on a deformed mesh $\mathscr{M}_n$. Following work of Sorkine et al. [116], Baran et al. [7] calculate the deformation gradient $\mathbf{T}_i$ by solving following weighted least-squares system, which minimises energy $E(\mathbf{T}_i)$, such that

$$E(\mathbf{T}_i) = \sum_{j \in \mathscr{N}_i} c_{ij} \left\| (\mathbf{p'}_i - \mathbf{p'}_j) - \mathbf{T}_i(\mathbf{p}_i - \mathbf{p}_j) \right\|^2 \ , \tag{2.5}$$

where $\mathscr{N}_i$ are indices of vertices in 1-ring neighbourhood of v_i and c_{ij} are cotangent weights calculated on $\mathscr{M}_0$. Note similarities to Equation 2.4.

Representing deformations as affine transformations $\mathbf{T}_i$ is not practical, because linear interpolation between shapes in this space is meaningless. Sumner et al. [121] propose to decompose $\mathbf{T}_i$ into rotational part and a scale/shear part using polar decomposition, so that $\mathbf{T}_i = \mathbf{R}_i\mathbf{S}_i$. Rotation matrix $\mathbf{R}_i$, mapped to $\log \mathbf{R}_i$ can be linearly interpolated and then converted back to $\mathbf{R}_i = \exp(\log \mathbf{R}_i)$. Linear interpolation of $\mathbf{S}_i$ is meaningful without additional conversions. This form of deformation representation is still variant to global rotations.

RIMD

Gao et al. [41] introduced a rotation-invariant mesh difference (RIMD). As opposed to Laplacian coordinates, RIMD is invariant in terms of both rigid rotations and translations, making it a suitable representation for training deep neural networks [122].

Deformation gradient is calculated as in Equation 2.5. Global rotations cancel out in RIMD representation, because instead of $\mathbf{R}_i$, only rotational difference $d\mathbf{R}_{ij} = \mathbf{R}_i^T \mathbf{R}_j$ is kept. This leads to a drawback of this representation - as $\mathbf{R}_i$ is not explicitly encoded in RIMD representation, a surface reconstruction requires a two-steps iterative process. Firstly, to calculate optimal positions $\mathbf{p}'$, $\mathbf{R}_i$ is initialised and fixed, and $\mathbf{p}'$ is solved in a least-squared sense using Cholesky factorisation. In the second step, $\mathbf{R}_i$ is explicitly obtained using Singular Value Decomposition. Due to lack of a closed form solution, surface reconstruction from RIMD is not computationally efficient.

ACAP

As-consistent-as-possible (ACAP) deformation representation is proposed by Gao et al. [42] as an improvement of computational issues of RIMD representation and its ability to handle large rotations. Deformation gradient is computed as in Equation 2.5. The rotation matrix $\mathbf{R}_i$ is represented using a rotation axis ω_i and rotation angle ϕ_i, so that all possible rotations can be encoded with $(\omega_i, \phi_i + t_i \cdot 2\pi)$, where t is an arbitrary integer. To ensure smooth, large rotations, integer $o_i \in \{-1, 1\}$ is introduced to indicate potential reversal of axis. Then, compatibility of axis orientations between adjacent vertices is maximised by maximising o_i and the rotation angle between adjacent vertices is minimised by minimising t_i.

Simplified deformation representation (DR)

ACAP representation optimises consistency of angles between adjacent vertices to handle rotations larger than 2π rad. However, in context of facial deformations, local rotations never exceed 2π rad. Wu et al. [136] address this fact and simplifies the deformation representation by skipping the optimisation of integers o_i and t_i. Wu et al. [136] prove effectiveness of the simplified representation in reconstructing caricatures of faces through combination and extrapolation of deformation features.

Skinning

Despite multiple approaches to representing spatial deformations [40], linear blend skinning (LBS) remains the most popular skinning model due to simple formulation and computational efficiency [76]. Next to blendshapes, it is primary representation of facial

deformations in animation and games industry. In LBS, surface deformation is represented as a set of bones $\mathscr{B} = \{\mathbf{R}_j|\mathbf{T}_j\}$, where $\mathbf{R}_j$ and $\mathbf{T}_j$ are rotation and transformation matrices of j^{th} bone. Position $\mathbf{p}_i$ of each vertex $v_i \in \mathscr{V}$ is transformed by weighted combination of bone transforms $\mathscr{B}$, where w_{ij} is the weight of j^{th} bone influencing v_i. The deformed vertex position

$$\mathbf{p'}_i = \sum_{j=1}^{|\mathscr{B}|} w_{ij}(\mathbf{R}_j\mathbf{p}_i + \mathbf{T}_j) \quad . \tag{2.6}$$

Smooth Skinning Decomposition with Rigid Bones (SSDR), introduced by Le and Deng [71], solves an inverse LBS problem. The algorithm automatically extracts bone-vertex weights map $\mathbf{W}$ and bone transforms $\mathscr{B}$ from a set of example poses $\mathscr{M}_n$ by minimising following energy using block coordinate descent algorithm:

$$E(w,\mathbf{R},\mathbf{T}) = \sum_{n=1}^{|\mathscr{M}|}\sum_{i=1}^{|\mathscr{V}|}\left\|\mathbf{p'}_i^n - \sum_{j=1}^{|\mathscr{B}|} w_{ij}(\mathbf{R}_j^n\mathbf{p}_i + \mathbf{T}_j^n)\right\|^2 \quad , \tag{2.7}$$

where $w_{ij} \geq 0$, $\sum_{j=1}^{|\mathscr{B}|} w_{ij} = 1$ and $\mathbf{R}_j^n$ is an orthogonal matrix with determinant = 1.

2.1.5 Voxels

Voxels are a 3D extension of a concept of pixels. Well established machine learning techniques in the field of 2D images can be easily applied to voxels due to their regular structure. On the other hand, data size of this representation limits the resolution of shapes because of high computational cost. It can be estimate that representing a face which meets requirements outlined in Section 1.2, requires 17.5 M voxels, leading to a 3D binary volume taking 140.6 MB of storage. For comparison, Euclidean coordinates mesh representation of the same face 3D model is 0.117 MB, and the deformation representation (DR) is 0.351 MB. Due to infeasible memory consumption, voxels representation is not considered in this research.

2.2 Generative Deep Learning on Euclidean Domains

2.2.1 Autoencoders

Dimensionality reduction of a large set of facial shapes is essential in devising a limited number of parameters which allow for facial shapes editing. Autoencoders aim at dimensionality reduction of data $\mathscr{X}$. They consist of two neural networks: encoder $E(\mathbf{x})$ which maps data samples $\mathbf{x} \in \mathscr{X}$ onto bottleneck $\mathbf{z}$, and decoder $D(\mathbf{z})$ which takes latent variable

$\mathbf{z}$ and outputs reconstructed data sample $\hat{\mathbf{x}}$. The networks $E(\mathbf{x})$ and $D(\mathbf{z})$ are optimised using gradient descent with an objective to minimise L2 norm between input $\mathbf{x}$ and output $\hat{\mathbf{x}}$. Such formulated autoencoders do not produce regular latent space and lead to overfitting, so arbitrary point $\mathbf{z}$ decoded by $D(\mathbf{z})$ will not generate new, meaningful content $\hat{\mathbf{x}}$.

To regularise the latent space of the autoencoders and, consequently, allow for aforementioned generative process to work, Kingma and Welling [69] introduced Variational Autoencoders (VAE). Instead of encoding an input as a single point, each $\mathbf{x}$ is encoded as a distribution $p(\mathbf{z}|\mathbf{x})$, which approximates a prior $p(\mathbf{z})$, so that encoder outputs mean μ and covariance σ of this normal distribution. In addition to reconstruction loss, KL divergence regularisation term is added to enforce distribution $p(\mathbf{z}|\mathbf{x})$ to be close to standard normal distribution. Thanks to regularisation, VAEs produce complete and continuous latent space of data $\mathscr{X}$ with generative properties. Training of Variational Autoencoders is very stable. When trained on images, their main drawback is blurry output. One of the reasons for the blurry output comes from the training principle, which is not able to assign low probability to blurry points (note that Generative Adversarial Networks from subsequent subsection solve this problem through adversarial training).

Tan et al. [122] create a Mesh VAE model of mesh deformations. They ignore non-Euclidean nature of 3D meshes and use only fully connected layers in both the encoder and decoder, following classical VAE architecture described in [69]. The Mesh VAE shows that standard Variational Autoencoders can be successfully utilised for both generation and interpolation of mesh deformations in RIMD representation [41]. However, due to large number of parameters and extensive memory consumption, fully connected layers are not practical in such models and can be replaced by more efficient graph convolutional layers, described in Section 2.3.1.

2.2.2 Adversarial training

Generative Adversarial Networks

Another popular generative model architecture, Generative Adversarial Network (GAN), was proposed by Goodfellow et al. [47]. GANs consist of two simultaneously trained networks, the generator G and the discriminator D. Generator G samples from normal Gaussian distribution and transforms the sample $G(\mathbf{z})$ so that it follows prior distribution. Discriminator D performs binary classification with a goal to classify all outputs from G as generated (false) distributions and all examples $\mathbf{x} \in \mathscr{X}$ from training dataset as real (true) distributions. The objective of generator G is to confuse the discriminator D and make it produce false positives. In such adversarial setup, D and G play a two-player min-max game and weights of the network G are implicitly optimised in iterative steps. In ideal

situation, optimisation of weights of G and D converges when G generates samples which follow the probability distribution of the prior and D is unable to differentiate between real and generated distributions.

When trained on images, unlike VAEs, GANs are able to produce crisp results. Nevertheless, training of original GANs is usually unstable and sampling tends to lack diversity. Very often, the generator learns to produce tiny set of outputs and traps the discriminator in a local minimum. Such failure is called *mode collapse*. To tackle unstable learning and poor mode coverage, multiple improvements over original GAN were proposed [108]. DCGAN [100] uses convolutional layers to improve visual quality of generated images. Instead of binary classifying discriminator, EBGAN [149] uses autoencoder discriminator with energy-based loss. WGAN [3] introduces alternative loss function based on the Wasserstein distance. It is further improved by WGAN-GP [49]. BEGAN [9] builds upon previous advancements and proposes autoencoder-based GAN which enforces equilibrium between generator and discriminator losses.

Progressive learning

Despite many improvements, previously mentioned approaches still struggle to generate larger images, because in higher resolution it is easier for the discriminator to differentiate between generated and training data, which leads to unstable training. Denton et al. [29] observe that it is easier to learn the mapping from latent space to high resolution images in steps, and use course-to-fine approach of LAPGAN, where cascade of convolutional GANs is trained within a Laplacian pyramid framework. Contrary to LAPGAN which consists of separate GANs for each level in the hierarchy, Karras et al. [63] propose a single, progressively growing GAN (PGGAN). Generator G and discriminator D are mirror images of each other, and as training progresses, higher resolution layers are added to both networks. This method not only stabilises, but also speeds up the training process.

Style-based approach

Given two 2D images: one representing arbitrary style, and another representing content, a neural network proposed by Huang and Belongie [58] is able to efficiently transfer that style onto given content. The novelty of the proposed network lies in Adaptive Instance Normalisation (AdaIN) layers, which align mean and variance of the input content to mean and variance of the given style. The authors demonstrate, that AdaIN performs style normalisation by normalising feature's statistics. This way, majority of the network manipulates the content of a given image, while AdaIN handles its style.

Seminal work of Karras et al. [64, 65], StyleGAN, incorporates AdaIN layers in generative part of GAN. When previously described GANs take the latent code as an

input to the first layer of the generator $\mathscr{G}$, in StyleGAN, the first layer is a learned tensor, which becomes constant during the inference. Instead, the latent code $\mathbf{z}$ is mapped onto intermediate latent space, which is input as a style into AdaIN at each convolutional layer. Moreover, in order to generate stochastic features, Gaussian noise is fed after each convolution. Visual comparisons show that noise highly impacts the generation of fine, stochastic details. StyleGAN is able to synthesise high-quality 2D images of faces with 1024^2 resolution. Moreover, thanks to latent code injected after each convolutional layer, the network provides scale-specific control over the generation process. Resulting latent code is disentangled, so that individual factors of variation consistently correspond with specific direction vectors in latent space.

Li et al. [78] propose a non-linear morphable model of faces which utilises StyleGAN architecture to generate colour texture and displacement maps with pore-level details. Variation in identity and facial expressions is represented through offsets in UV space. The correlation between textures and geometry is preserved through their joint training. The output from StyleGAN is further upscaled to 4K using super-resolution network introduced by Ledig et al. [72].

Hybrid models

Variational Autoencoders and Generative Adversarial Networks are the most popular deep generative approaches. Training of VAEs is very stable, yet decoded images lack details and tend to be blurry. Although training stability of GANs was substantially improved, they can be challenging to train. On the other hand, GANs produce more detailed, sharper images. Having these in mind, Huang et al. [57] propose to combine both training methods in their IntroVAE model. Most recently, Pidhorskyi et al. [94] introduce StyleALAE, which leverages advantages of GANs, VAEs and AdaIN. The progressively growing encoder network $\mathscr{E}$ outputs style vector after each of its Instance Normalisation (IN) layer. The style is input to Adaptive Instance Normalisation (AdaIN) layers of the corresponding, progressively growing, adversarially trained generator $\mathscr{G}$. StyleALAE is especially interesting for us: due to presence of the encoder, it not only synthesises high-quality 2D images of faces, but also serves as their disentangled representation with multi-scale control.

2.2.3 Denoising diffusion models

Recent developments in denoising diffusion models demonstrate outstanding performance in generative modelling. For example, Dhariwal and Nichol [31] employ their ablated diffusion model (ADM) to outperform state-of-the-art GANs on image-synthesis task. Notable examples of text-guided generative models include Stable Diffusion, which builds

upon work of Rombach et al. [103], DALL·E 2 [101], and GLIDE [91], the diffusion models conditioned on CLIP [99] embeddings. Croitoru et al. [26] provide a thorough literature survey on diffusion models in computer vision.

Diffusion models have been applied to a wide variety of generative and discriminative tasks on 2D images. In the area of 2D facial registration, DiffuseMorph [67] adapts diffusion models to generate deformation fields.

Denoising Diffusion Probabilistic Models (DDPM)

Denoising Diffusion Probabilistic Models (DDPMs) have been proposed in [114, 55]. They are a class of generative models which gradually refine noise through a learned denoising process.

In the forward process, the noising is performed as follows:

$$p(x_t \mid x_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t} \cdot x_{t-1}, \beta_t \cdot \mathbf{I}), \forall t \in \{1, ..., T\}. \tag{2.8}$$

The original sample x_0 is corrupted by the Gaussian noise in T iterations. Given the previous sample x_{t-1}, the distribution of the next sample x_t is the normal distribution of mean and co- variance calculated according to the variance schedule $\beta_1, ..., \beta_T \in [0, 1)$. For example, Ho et al. [55] choose $(\beta_t)_{t=1}^T$ to linearly increase from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$, where $T = 1000$.

Equation 2.8 calculates the probability distribution of x_t given the previous noising step, i.e. in a recursive process. It can be rewritten to explicitly calculate the distributions of any x_t given original x_0:

$$p(x_t \mid x_0) = \mathcal{N}(\sqrt{\hat{\beta}_t} \cdot x_0, (1-\hat{\beta}_t) \cdot \mathbf{I}). \tag{2.9}$$

The mean and covariance terms use $\hat{\beta}_t = \prod_{i=1}^{t}(1-\beta_t)$, i.e. a product of all $(1-\beta_t)$ up to desired noising step t. Based on Equation 2.9, x_t can be sampled simply by adding the mean term to the product of a standard Gaussian distribution sample z_t and the standard deviation (square root of covariance):

$$x_t = \sqrt{\hat{\beta}_t} \cdot x_0 + \sqrt{1-\hat{\beta}_t} \cdot z_t, z_t \sim \mathcal{N}(0, \mathbf{I}). \tag{2.10}$$

There are at least two approaches to the reverse process. They both begin with a sample from a standard Gaussian distribution $x_T \sim \mathcal{N}(0, \mathbf{I})$, and iteratively revert the noising process to finally obtain a new, generated sample x_0:

$$p(x_{t-1} \mid x_t) = \mathcal{N}(\mu(x_t, t), \Sigma(x_t, t)). \tag{2.11}$$

While the forward process iteratively removes information through noising, the reverse process starts from noise and iteratively generates information.

In the first approach, a neural network takes as an input sample x_t and embedding at time step t. This embedding is devised from positional encoding, which describes the position of a sample in a sequence, so that each position is assigned a unique representation. The network is trained to predict the mean μ and covariance Σ of the normal distribution. This approach can be turned into objective of maximising evidence lower-bound (ELBO), where at each time step t, $p_\theta(x_t - 1 \mid x_t)$ is as close as possible to the true posterior in the forward pass. Kullback–Leibler (KL) divergence is used to compare these two distributions.

$$p(x_{t-1} \mid x_t) = \mathcal{N}(\mu(x_t,t), \Sigma(x_t,t)). \tag{2.12}$$

The second approach simplifies the learning objective and Nichol and Dhariwal [90] propose to parametrise $\Sigma_\theta(x_t,t)$ and combine both objectives into a single loss function $L_{hybrid} = L_{simple} + \lambda L_{vlb}, \lambda = 0.001$. They also apply a stop-gradient to the $\mu_\theta(x_t,t)$ for the L_{vlb}, so that L_{vlb} can guide $\Sigma(x_t,t)$ and L_{simple} has the most influence on $\mu_\theta(x_t,t)$. According to Saharia et al. [105], L_1 norm leads to lower sample diversity than originally used L_2 norm [55], but it reduces potential hallucinations.

Conditional diffusion

Conditioning is used to guide a generative process. Here, two conditioning methods are considered, concatenation and cross-attention.

The generative process begins with Gaussian noise $y_T \sim \mathcal{N}(0, \mathbf{I})$ and it is iteratively refined in T steps, at each step learning conditional parametric distribution $p_\theta(y_{t-1} \mid y_t, x), \forall t \in \{1, ..., T\}$, where θ are learnable parameters of the model p_θ. At each step, y_{t-1} and x are concatenated to form an input to p_θ. Additional information can be used as well, for example Saharia et al. [106] concatenate noise statistics to help with the denoising process. Data must be spatially aligned before concatenation. Therefore, concatenation of different modalities can be difficult. Even within the same modality, lower resolution images guiding higher resolution generation must be upscaled through interpolation to match high-resolution dimensions.

Rombach et al. [103] propose an alternative conditioning mechanism using cross-attention. The main advantage of cross-attention over previously described concatenation is flexibility. This mechanism allows to mix latent codes of different modalities. Attention is permutation invariant. It can be beneficial in domains where order of elements is irrelevant. For example, a set of photographs of an object from various angles or an unindexed pointcloud. In permutation-variant domains, such as latent codes, images or meshes, data must be positionally encoded. There are multiple choices of positional

encodings, both learned and fixed. An input to Scaled Dot-Product Attention [128] consists of query-key-value (Q, K, V):

$$\text{Attention}(Q, K, V) = \text{softmax}\Big(\frac{QK_T}{\sqrt{d}}\Big) \cdot V \tag{2.13}$$

In self-attention, Q, K and V are devised from the same input array. In contrast, cross-attention calculates Q from the first array, and K and V from the second array. The output has dimensions of the first array, independently from dimensions of the second array.

2.3 Geometric Deep Learning

Undirected graph (mesh) is the most common representation of a facial shape. Deep neural networks have proven to be powerful tools in extracting latent representations of data in Euclidean domain. In recent years, these deep learning approaches extended to irregular graphs [139, 15, 140] and allowed for learning non-linear, lower-dimensional embeddings of meshes, including human faces.

2.3.1 Convolutional Graph Neural Networks

The concept of graph convolutions generalises traditional operation of a convolution on structured grids, such as 2D images or 3D voxels. Convolutional layers aim at extracting representation of vertex $v_i \in \mathcal{V}$ by aggregating feature $\mathbf{f}_i$ and its neighbours. Such approach reduces the number of parameters in the model and helps the network to learn local patterns.

Graph signal processing

Graph signal processing (GSP) extends classical signal processing tools, such as Fourier transform or filtering, to signals residing on a set of vertices of a graph [92]. Convolutional filters on a graph need to satisfy two main properties: they can be applied symmetrically to each vertex and they aggregate information in arbitrary neighbourhood of this vertex. Satisfying these conditions in spatial domain is not straightforward. Representing data defined on a graph in frequency domain helps to establish such convolving filter.

Graph Laplacian (see Equation 2.4) can be used to convert signal residing on vertices of an undirected graph to frequency domain. The normalised graph Laplacian $\hat{\mathbf{L}}$ is

$$\hat{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \quad . \tag{2.14}$$

$\hat{\mathbf{L}}$ can be decomposed (using SVD) into $\hat{\mathbf{L}} = \mathbf{U}\Lambda\mathbf{U}^T$, where $\mathbf{U}$ are eigenvectors of $\hat{\mathbf{L}}$, which form Fourier basis of a frequency space. Λ is a diagonal matrix of eigenvalues of $\hat{\mathbf{L}}$, representing different frequencies. This way, both, a signal $\mathbf{x}$ residing on vertices and a convolutional filter $\mathbf{g}$, can be projected onto diagonal matrices in Fourier domain using graph Fourier transform (GFT), transforming signal $\mathbf{x}$ into $\hat{\mathbf{x}} = \mathbf{U}^T\mathbf{x}$. To recover original signal $\mathbf{x}$, inverse GFT is applied: $\mathbf{x} = \mathbf{U}\hat{\mathbf{x}}$. Knowing that convolving two signals in spectral domain means multiplying them together, the operation of convolution of signal $\mathbf{x}$ with filter $\mathbf{g}$ is formulated as

$$\mathbf{x} *_G \mathbf{g}_\theta = \mathbf{U}\mathbf{g}_\theta\mathbf{U}^T\mathbf{x} \ , \tag{2.15}$$

where $\mathbf{g}_\theta = \text{diag}(\mathbf{U}^T\mathbf{g})$, and $*_G$ is a graph convolution operator.

Spectral graph convolutions

The first formulation of spectral convolutions on graphs in the field of machine learning comes from Bruna et al. [17]. They define a convolutional operator in the same way as in Equation 2.15, with $\mathbf{g}_\theta = \Theta$ being a set of learnable parameters. There are three major disadvantages of this formulation. Firstly, the number of parameters is strictly dependent on $|\mathcal{V}|$. For dense graphs, large number of parameters can result in overfitting. Secondly, convolutional operator is defined in terms of $\mathbf{U}$, which is expensive to calculate on larger graphs. Thirdly, convolutional filter is not localised, i.e. it is not defined on a small neighbourhood around a central vertex.

To overcome aforementioned issues, Defferrard et al. [27] propose ChebNet network, which approximates filter $\mathbf{g}_\theta$ by the Chebyshev polynomials of Λ, so that convolutional operator is redefined as

$$\mathbf{x} *_G \mathbf{g}_\theta = \mathbf{U}\left(\sum_{i=0}^{K}\theta_i T_i(\tilde{\Lambda})\right)\mathbf{U}^T\mathbf{x} = \sum_{i=0}^{K}\theta_i T_i(\tilde{\mathbf{L}})\mathbf{x} \ , \tag{2.16}$$

where $\tilde{\Lambda} = 2\Lambda/\lambda_{max} - \mathbf{I}$ and $\tilde{\mathbf{L}} = 2\hat{\mathbf{L}}/\lambda_{max} - \mathbf{I}$, K is an order of Chebyshev polynomials defined recursively by function $T_i(\cdot)$, $\theta \in \mathbb{R}^K$ is polynomial coefficients and λ_{max} is the largest eigenvalue. Formulation of a filter in ChebNet avoids expensive computation of basis $\mathbf{U}$ and significantly reduces the number of learnable parameters. Moreover, convolving filter approximated with Chebyshev polynomials is localised in K-ring neighbourhood.

MeshVAE model of Tan et al. [122] explores probabilistic latent space of 3D shape deformations using fully connected layers, which results in vast number of learnable parameters. As an improvement, Yuan et al. [145] use convolutional layers with the same operator as in Equation 2.16 to convolve pre-processed ACAP deformation features

residing on vertices $\mathscr{V}$ of a mesh $\mathscr{M}$. Consequently, their model is able to handle dense meshes and it reduces a chance of data overfitting.

Kipf and Welling [70] further decrease the number of learnable parameters using first-order approximation of ChebNet. They assume $K = 1$, $\lambda_{max} = 2$, $\theta = \theta_0 = -\theta_1$, add self-loops to adjacency matrix and normalise it. This approach is effective in vertex classification tasks, however, to our knowledge, its performance on representation learning of 3D deformations has not been explored yet.

Spatial graph convolutions

More recent approaches move away from isotropic convolutional operators in spectral domain in favour of the anisotropic ones, defined in the spatial domain. Bouritsas et al. [14] and Gong et al. [46] build graph convolutional operators which explicitly enforce consistent ordering of vertices using a spiral. These approaches ignore irregularity of graphs. In contrast, in [43] and [151], convolutional operators learn adaptive weighting matrices. Additionally, the convolutional operators by Zhou et al. [151] support transpose convolutions. All the aforementioned convolutional operations are applicable to features residing on vertices.

So far, all the presented convolutional operations were applicable to features residing on vertices $\mathscr{V}$. Yet, features can be defined on edges or faces as well. Hanocka et al. [50] introduce a convolution operator $\mathbf{g}$ which aggregates feature $\mathbf{x}_0$ of a central edge and features $\mathbf{x}^j$ of its four adjacent edges:

$$\mathbf{x}_0 \cdot \mathbf{g}_0 + \sum_{j=1}^{4} \mathbf{g}_j \cdot \mathbf{x}_j \ , \tag{2.17}$$

where $\mathbf{g}$ is a convolutional filter. As features $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ edges $\{e_1, e_2, e_3, e_4\} \in \mathscr{E}$ can be ordered in two different ways, a symmetric function is applied to them, so that convolution invariance is ensured.

To benefit from aforementioned convolutional operator, invariant deformation representations, such as DR or ACAP, would have to be redefined in terms of edges. Interestingly, RIMD representation already encodes rotation differences $d\mathbf{R}_{ij}$ residing on half-edges.

2.3.2 Pooling and de-pooling

Pooling is an operation which sub-samples a feature map to reduce computational load, memory consumption, as well as the number of learnable parameters. When applied to subsequent convolutional layers, it allows to train multi-scale convolutional filters, capturing local and global patterns. Pooling has been widely used in deep networks for 2D image processing and recently it was generalised to graphs in various ways. There exist

numerous graph coarsening methods applied to generic directed and undirected graphs, however this overview will be focused specifically on pooling and de-pooling operations on 3D meshes.

In domain of meshes, pooling can be performed through classical mesh simplification methods, such as the algorithm by Garland and Heckbert [44], which iteratively contracts pairs of vertices that maintain surface error approximations using quadric matrices. Ranjan et al. [102] design graph pooling and de-pooling operations based on this simplification algorithm. As down-sampling of surfaces is lossy, they perform pooling and build de-pooling matrices at the same time. It is achieved by projecting original vertices onto closest triangle in down-sampled mesh and storing their positions in barycentric coordinates.

Yuan et al. [145] also use mesh simplification [44] to devise mesh pooling operations. Unlike previous method, their approach produces meshes with evenly distributed triangles (due to an additional error term which penalises length of the longest edge around each vertex). Simplification is performed through edge contraction, half-way the edge length. Average of features residing on contracted vertices is assigned to a newly created vertex. Accordingly, max-pooling could be implemented by assigning maximal feature to the new vertex. Similarly to image-based pooling, Yuan et al. [145] propose to assign features on vertices of the simplified mesh to corresponding vertices on the dense mesh.

Hanocka et al. [50] introduce pooling based on edge collapse, where an order of edges to collapse is dependent on the magnitude of the features residing on these edges. It enables non-uniform deletion of areas which are not useful for minimising particular loss in the neural network. As in other methods, vertex positions are recorded during the pooling operation and retrieved in the de-pooling. Task-dependent pooling achieved state-of-the-art results in mesh classification problems, however it was not applied to mesh generation tasks, because in this type of pooling shape of the surface is not preserved.

2.3.3 Spectral Mesh Processing

Various methods of spectral mesh processing have been developed to address the problems of shape analysis, mesh simplification, surface segmentation, and shape correspondence. Sorkine [115] and Zhang et al. [147] comprehensively review existing work until 2005 and 2010, respectively. Recent work in the field is reviewed in the following paragraphs.

Melzi et al. obtain smooth, local, controllable, orthogonal, and efficient basis called Localized Manifold Harmonics (LMH) through the spectral decomposition of new types of intrinsic operators. The authors integrate local details provided by the basis and the global information from the Laplacian eigenfunctions to deal with local spectral shape analysis [83]. Xu et al. fit the original 3D model with a finite subdivision surface and restrict the eigenproblem with a subdivision linear subspace to obtain intrinsic shape information

of 3D models through fast calculation of large-scale Laplace–Beltrami eigenproblem on the models [142]. Lescoat et al. propose a new mesh simplification algorithm, which uses a spectrum-preserving mesh decimation scheme to simplify input meshes and makes the Laplacian of simplified meshes spectrally close to the Laplacian of input meshes [75].

Wang et al. use a Laplacian operator and select and combine some of its sub-eigenvectors to compute a single segmentation field to conduct mesh segmentation [131]. Tong et al. build a Laplacian matrix, propose a spectral mesh segmentation method, which converts mesh segmentation into an ℓ_0 gradient minimisation problem, and devise a fast algorithm to solve the minimisation problem [125]. Bao et al. devise a feature-aware simplification algorithm to create a coarse mesh. Then, they use the spectral segmentation method proposed in [125] to perform partition on the coarse mesh to obtain a coarse segmentation. Next, they reverse the simplification process to map the coarse mesh to the input mesh and smooth jaggy boundaries to develop a spectral segmentation method for large meshes [5].

Jain and Zhang first transform two 3D meshes into spectral domain and find and match the embeddings of the two 3D meshes in the spectral domain to obtain vertex-to-vertex correspondence between the two 3D meshes [61]. Dubrovina and Kimmel use the eigenfunctions of the Laplace–Beltrami operator to calculate surface descriptors and match surface descriptors to develop a correspondence detection method [33]. Melzi et al. introduce iterative spectral upsampling to obtain high-quality correspondences with a small number of coefficients in the spectral domain [82].

Spectral methods share a common framework. They define a matrix representing a discretisation of a continuous operator over a mesh. In this thesis, the concept of spectral mesh processing in introduced to 3D shape representation learning.

2.3.4 Geometric Deep Learning in Spectral Domain

Dong et al. propose a convolutional neural network framework called Laplacian2Mesh by mapping 3D meshes to a multi-dimensional Laplacian–Beltrami space to deal with irregular triangle meshes for shape classification and segmentation [32]. Lemeunier et al. integrate spectral mesh processing and deep learning models consisting of a convolutional autoencoder and a transformer to develop a spectral transformer called SpecTrHuMS to generate human mesh sequences [74]. Qiao et al. present a deep learning approach that uses Laplacian spectral clustering to build a fine-to-coarse mesh hierarchy and integrate Laplacian spectral analysis and mesh feature aggregation blocks to encode mesh connectivity for shape segmentation and classification [97]. Based on the idea of approximating low and mid frequencies on coarse grids, Nasikun and Hildebrandt investigate a new solver called the Hierarchical Subspace Iteration Method that can solve sparse Laplace–Beltrami

eigenproblems on meshes faster than existing methods based on Lanczos iterations, preconditioned conjugate gradients, and subspace iterations [88].

2.4 Parametric Face Models

Parametric models of facial shapes [34] aim at learning the general representation of faces that captures prior knowledge about them. This prior knowledge helps to resolve ambiguities in otherwise ill-posed computer vision tasks, including 3D face reconstruction from images covered in Section 2.5.

2.4.1 Linear models

Global 3DMM

The pioneering 3D Morphable Model (3DMM) proposed by Blanz and Vetter [11] is a probability distribution function, which measures the likelihood that a set of input coefficients will output plausible 3D face shape. It uses Principal Component Analysis (PCA) to reduce dimensionality of 200 facial identity shapes and corresponding 2D textures. Basis vectors $\mathbf{U} = \{\mathbf{b}_1^T, ..., \mathbf{b}_n^T\}$ and standard deviation coefficients $\text{diag}(\Lambda) = \{\lambda_1, ..., \lambda_n\}$ resulting from PCA are used to reconstruct or generate facial shapes as

$$\mathbf{p} = \bar{\mathbf{p}} + \sum_{i=1}^{n} \alpha_i \mathbf{b}_i \ , \tag{2.18}$$

where $\mathbf{p}$ is a vectorised representation of 3D vertex positions of the reconstructed shape, $\bar{\mathbf{p}}$ is the mean shape and α is the vector of parameters. Analogical formulation applies to the model of facial textures. Given that a few tens of first basis vectors $\mathbf{b}_i$ are enough to represent almost 90% of the original variance, dimensionality reduction is achieved by ignoring least significant vectors and, consequently, limiting the number of parameters α. The concept of a 3DMM [11] has stood the test of time and improved over the years. For example, Booth et al. [13] build LSFM - the largest face 3DMM model using 9663 identities, and prove its regularisation effectiveness in 3D face reconstruction from images [12].

Local models

Global 3DMMs [11, 13] are very compact representations of facial shapes, thus require relatively small number of parameters. Nonetheless, they struggle to reconstruct local details of distinct facial areas. Moreover, bases of global 3DMMs lack a semantic interpretation. To overcome these limitations, researchers either manually segment the face and

learn separate PCA models of each region [11], use automated segmentation process [123], or perform Independent Component Analysis to obtain a sparse model [89]. In contrast to previous approaches, Wu et al. [134] proposes to represent local displacements with a set of patches and regularise overall facial shape using the underlying anatomical bone structure. This approach requires facial shapes with corresponding jaw and skull bones data, which are very difficult to obtain (through either magnetic resonance imaging (MRI) or computerised tomography (CT) scans). Consequently, approaches which require MRI or CT scans are infeasible for our research.

Bilinear and multilinear models

Aforementioned techniques either encode identity shapes alone, or entangle them with expressions. Resulting models do not allow for independent manipulation of either expressions or identities. Numerous authors tackle this issue with a bilinear model, where two PCA models are combined in additive manner. A different variation of this bilinear model is used by Thies et al. [124], whose model encodes identity shapes in space with orthogonal bases from PCA, while expressions are encoded in space with blendshapes as the bases. This approach does not results in compact representation of facial shapes, because blenshapes do not form an orthogonal space.

Although bilinear models disentangle identites from expressions, they do not preserve any correlations between them. In order to reestablish these correlations, Vlasic et al. [130] proposes a multilinear model that performs higher-order singular value decomposition (HOSVD) on a 4-tensor, where identities, expressions, visemes and vertex coordinates are the separate modes. Such formulation preserves correlations between modes, but is limited to complete tensors where each identity has corresponding set of exactly the same facial expressions.

Skinned linear models

While linear models are capable of compact representations of deformable shapes, they are not used to characterise articulated poses. Loper et al. [81] propose a Skinned Multi-Person Linear (SMPL) model of human body poses, which combines linear blend skinning (see Section 2.1.4) with identity and pose-dependent corrective shapes PCA models.

Li et al. [79] introduce Faces Learned with an Articulated Model and Expressions (FLAME), which extends SMPL method to 3D heads. In FLAME, linear blend skinning is used to represent articulated poses of the neck, jaw, and eyeballs. The mean identity template mesh is added to weighted PCA vectors representing head shape, pose correctives and expression offsets. The resulting mesh is deformed using a skinning function. Joint locations depend on face identity. To model this relationship, FLAME formulates a joints

predictor, which is a learned, sparse matrix. FLAME remains one of the most popular parametric representations of 3D human faces.

2.4.2 Non-linear models

Linear models assume Gaussian prior distribution of facial shapes. This assumption is not correct [34]. Non-linear models have potential to generalise to wider range of unseen examples and provide more compact representation. In recent years, deep learning approach to generating non-linear models of facial shapes and textures became the area of active research.

Models in 2D image space

Deep generative models on regular grids, such as images, are strikingly successful in approximating prior distributions of faces . Recently, researchers begun applying these 2D generative approaches to learning the manifold of facial displacements. In order to directly apply deep learning architectures in Euclidean domain, facial displacements need to be represented as either 2D images in UV space or voxels . Bagautdinov et al. [4] use Variational Autoencoders to learn displacement maps. As VAEs often produce blurry results, they propose a compositional approach, where different level of detail is learned on each layer of the encoder, and the decoder combines these multi-level representations into the final displacement map. Generative Adversarial Networks are especially popular choice of architecture [45, 111] and lead to high-quality results [78]. Abrevaya et al. [1] bypass challenges of applying traditional deep learning methods to meshes by using only fully-connected layers. Their method is based on GAN with generator producing 3D facial meshes in Euclidean space, and the discriminator evaluating these shapes in UV space. Slossberg et al. [113] demonstrate strong correlation between facial identity shapes and their colour textures. They propose to use a progressively-growing GAN to generate colour textures of faces and, based on these textures, optimise coefficients of a 3DMM (as in Equation 2.18) to obtain facial identity shapes. Li et al. [78] propose the model able to generate facial identity and expression shapes at pore-level resolution, coupled with multiple 4K texture maps. The method utilises StyleGAN architecture to learn displacements in UV space. Furthermore, the super-resolution network [72] is used to up-sample texture maps.

Aforementioned methods, projecting facial displacements onto 2D images, benefit from well-established generative models in Eucleadean domain. Nonetheless, they require a large number of learnable parameters, which leads to high computational cost and risk of overfitting. Most recently, alternative approaches attempt to learn representations of facial shapes directly on 3D meshes using geometric deep learning.

Models in 3D mesh space

Work of Ranjan et al. [102] is the first geometric learning approach to representing facial shapes. It utilises VAE architecture with custom pooling layers and graph convolutional operations described in Equation 2.16. Models [14, 22, 43, 151, 129] improve this architecture with custom convolutional and aggregation operators. Cheng et al. [23] introduce MeshGAN, the first adversarial approach to learning facial manifold directly on 3D meshes. It integrates graph convolutional operator based on Chebyshev polynomials and uses neutral architecture based on BEGAN [9]. In addition to shape, Zhou et al. [150] jointly learn deformation offset and colour residing on vertices of a face mesh.

All above methods represent facial displacements in Euclidean coordinates - a variant representation, which performs worse than invariant alternatives [122, 145]. In contrast, Jiang et al. [62] learn disentangled manifolds of facial identities and expressions in deformation representation (DR), described in Section 2.1.4. They introduce a framework consisting of graph VAEs, which disentangle facial mesh into identity shape and the expression on a mean shape. Finally, the fusion network combines identity and expression and reestablishes correspondence between them.

Apart from geometric deep learning methods, various physically-based facial models are introduced [60]. Several authors propose highly-expressive local models, further constrained with underneath skull and jaw bone [48, 134]. Such approaches allow for anatomical control, where indirect facial surface editing is achieved through manipulation of underlying skull and/or muscles. Although anatomically constrained models effectively regularise local models, they require not only high-quality facial surface scans, but also corresponding bone structures. Such paired data is difficult and expensive to obtain.

Neural scene representations

More recently, Neural Radiance Fields (NeRF) and 3D Gaussian Splatting (3DGS) gained popularity in 3D scene representation, including human heads. Rabby and Zhang [98], Chen and Wang [20] and Bao et al. [6] provide detailed surveys of NeRF and 3DGS.

NeRF [85] reconstructs a 3D scene from a set of 2D images taken from different viewpoints. By training a neural network to understand the relationship between the scene's geometry and appearance, NeRF can generate highly photorealistic renderings from any perspective. During training, the network learns to map 2D image coordinates and viewing directions to corresponding 3D points and colours. This is achieved by optimising the network to minimise the difference between rendered and real images. Finally, the trained network can synthesise new views by querying the radiance field at different points in 3D space, allowing for dynamic scene rendering. Such implicit radiance field function can be formulated as a multi-layer perceptron map $\mathrm{MLP}(x, y, z, \theta, \phi)$, which maps (x, y, z)

coordinates and viewing directions (θ, ϕ) onto colour and density values. This approach comes at a high computational cost due to volumetric ray marching rendering.

3D Gaussian Splatting [66] is a hybrid 3D representation that combines the advantages of explicit and implicit radiance fields by using learnable 3D Gaussians to model a scene. Unlike NeRF, Gaussian Splatting retains structured data storage while benefiting from neural optimisation techniques. This approach allows for real-time, high-quality rendering with reduced training time. The 3D Gaussian representation is mathematically defined as a weighted sum of Gaussian functions $\sum_i G(x, y, z, \mu_i, \Sigma_i) \cdot c_i(\theta, \phi)$, where each Gaussian is characterised by a mean μ_i, a covariance matrix Σ_i, and a view-dependent colour c_i. Rendering of 3DGS involves projecting the learnable 3D Gaussians onto a 2D image plane in a process known as "splatting." This step maps the 3D representation onto a pixel-based view. The Gaussians are then sorted based on their depth to maintain correct visibility ordering. Finally, the pixel values are computed by blending the contributions of overlapping Gaussians, resulting in a high-quality, photorealistic image. Compared to NeRF, this approach significantly improves rendering speed and reduces computational cost.

NeRF and 3DGS are applied to representation of 3D avatars [21, 86, 56, 30, 96, 107, 152]. In 3D head reconstruction, many approaches leverage FLAME model (see Section 2.4.1) as geometric prior for 3D Gaussian Splatting. In these methods, FLAME acts as a coarse guidance for 3DGS reconstruction.

2.5 Monocular 3D Face Reconstruction

The literature on 3D face reconstruction is vast and beyond the scope of this review. Comprehensive survey can be found in the report by Zollhöfer et al. [153]. Given that Objective 2 is to develop a parametric face model, which improves the facial reconstruction results, a brief overview of different 3D face reconstruction methods is provided, with particular emphasis on statistical models acting as 3D face priors.

2.5.1 Structure-from-Motion

The 3D structure of rigid scenes can be reconstructed from a series of images and 2D landmarks. Such pipeline, referred to as Structure-from-Motion (SfM), requires controlled environment of acquisition and, on its own, is insufficient in 3D face reconstruction from in-the-wild images, as demonstrated in [53]. Combination of SfM with other visual cues can lead to high-quality results, such as those of Agrawal et al. [2]. They generate a point cloud using SfM method, and then reconstruct high-quality, static 3D faces using points, silhouettes and landmarks as constraints.

2.5.2 Other shape cues from 2D images

As mentioned above, unconstrained SfM fails to produce satisfactory results. In order to constrain the outputs, various cues are extracted from 2D images, such as landmarks, edges, silhouettes, textures and shading.

For example, to reconstruct diverse and exaggerated 3D caricatures from their 2D representations, Zhang et al. [148] propose to weakly-constrain their model with landmarks. Additionally, they utilise extrapolation properties of ACAP deformation representation, which helps to obtain highly-exaggerated facial shapes.

Shape-from-shading (SfS) method extracts facial surface details from 2D images using image formation assumptions, such as Lambertian reflectance. SfM is able to reconstruct fine facial features, such as wrinkles [133]. Nevertheless, it relies on difficult to estimate lighting and shading information, which leads to many unwanted artefacts. Besides that, this method on its own is able to produce only 2.5D facial shapes [137] and requires more constraints to obtain a full 3D face model.

2.5.3 Statistical model priors

Most 3D face reconstruction methods regularise shapes with statistical models, which are covered in Section 2.4. Parametric face models are able to correctly reflect global shape of an individual. Nevertheless, their resolution is often limited and they tend to lack details such as wrinkles. The state-of-the-art methods attempt to balance two conflicting effects: (1) regularisation of the parametric models, which accurately reconstructs global shape and (2) constraints from cues, which bring fine-level details.

2.5.4 Estimation of the parameters

Parameters of the parametric face model need to be estimated based on a 2D image or a sequence of images. There are two main approaches to devise the parameters of the statistical model. The first one, analysis-by-synthesis, is an optimisation process, where a synthetic image is generated from a set of parameters which are iteratively refined to minimise the difference between this generated image and a reference image. Booth et al. [12] propose a generative approach to monocular 3D face reconstruction, which is able to handle in-the-wild images. The major advantage of the analysis-by-synthesis is ability to perform parameters optimisation without supervision with ground-truth 3D meshes, which are inaccessible for such images captured in-the-wild. The second technique is a discriminative approach, which regresses the parameters of the model directly from an image or set a of images. With rapid development of the deep convolutional networks,

this approach can effectively devise parameters of the model in less time than the iterative optimisation.

Lin et al. [80] take the discriminative approach to reconstruct 3D shape and coarse RGB texture, which is further refined using graph convolutional operations, as in Equation 2.16 with pooling layers as in [102]. This approach is particularly interesting, as it learns RGB colour values residing on vertices.

Several methods allow for multi-view input [135, 109, 28]. The last one, regresses parameters of a 3DMM in unsupervised manner, with a set of constraints, including skin-detecting mask and a pre-trained face recognition network. The multi-view input is possible thanks to image-quality measure which provides weights for individual shapes aggregation.

A coarse-to-fine framework, where general shape and fine details are generated in different stages, is used to balance contradicting effects of strong regularisation and variance [146, 126, 19]. In [146], facial shapes are represented with depth image, so the method does not allow for full 3D head reconstruction. In [126] and [19], the overall shape is reconstructed with fitted 3DMM, and fine details are represented with a bump and displacement maps respectively. These methods are able to reproduce detailed likeness of an individual, yet due to noise and multiple unwanted artefacts, their outputs can not be used directly in VFX, games and animation production.

CHAPTER3

COMPARISON OF SHAPE REPRESENTATIONS IN DEEP 3D MORPHABLE MODELS

3.1 Introduction

The parametric face models are extensively used in computer vision tasks. The literature review from Chapter 2 indicates that building upon and improving deep 3D morphable models (Deep3DMMs) can be helpful in meeting the objectives outlined in Chapter 1. The application of deep 3D morphable models in high-end visual effects and AAA games production is limited, as these models struggle to reconstruct facial meshes with minimal disparity, capturing both local details and the global shape.

This chapter investigates the effects of using different input and output representations to Deep3DMMs and the standardisation and normalisation of these representations, to improve the perceptual quality and geometric accuracy of facial meshes output from these models. The Deep3DMM Comparison Platform is implemented to efficiently perform these comparisons on multiple deep 3D morphable models in the industrial setting, with different datasets and 3D face representations. Findings from this chapter are used in methods described in Chapters 4 and 5.

3.1.1 Inspiration by the quantisation of different mesh representations

An observation of Sorkine et al. [119] is particularly relevant in this work. They focus on suppressing the visual effects of quantisation. The authors demonstrate that:

1. Quantisation of meshes in differential representation introduces perceptually insignificant, low-frequency error.

2. Quantisation of 3D meshes in 3D Euclidean coordinates space introduces noticeable, high-frequency error.

Differential representation explicitly encodes the surface properties of the mesh, and the position of vertices is encoded implicitly in this representation. In contrast, 3D Euclidean coordinates describe the position of vertices in explicit form, and the surface properties can be implicitly derived from this representation. It can be concluded that the properties explicitly encoded in a representation are best preserved after that representation is subjected to quantisation. Conversely, the properties which are only implicitly encoded in a representation can be more severely affected by quantisation. Since encoding 3D shapes in a compact parametric space is lossy, deep 3D morphable models have similar compression side-effects to quantisation.

Based on these observations and the following discussion, it can be hypothesised that:

1. Using different input and output representations improves the state-of-the-art facial mesh reconstruction results in terms of either perceptual quality or geometric accuracy.

2. Using input and output representations, which explicitly encode the surface properties, improves the perceptual quality of the resulting meshes.

3. Using input and output representations, which explicitly encode the vertex positions in 3D space, improves the geometric accuracy of the resulting meshes.

3.1.2 Impact of the model, input representation and its preprocessing on the quality of the outputs

Most previous publications on deep 3D morphable models, as outlined in Chapter 2, evaluate the proposed methods against only one mesh representation, most commonly standardised Euclidean coordinates. Consequently, the performance of these models with other mesh representations is unknown, which provides an opportunity to evaluate these models from the representation perspective. Furthermore, it allows to distinguish the combinations of models and representations which outperform the results achieved in the original implementations of these models.

In the aforementioned publications, evaluation metrics that compare ground truth meshes with reconstructed meshes are usually limited to Euclidean distance. However, as demonstrated in Chapter 2, L-norm metrics poorly correlate with the perceptual quality of

reconstructed meshes. Given that the perceptual quality of a mesh is an essential factor in assessing the method's usability in visual applications, the lack of perceptual evaluation of deep 3D morphable models is a noticeable gap in the literature.

It is proposed to design and implement the Deep3DMM Comparison Platform, which accommodates the preprocessing of various datasets to different input representations and training multiple models within the same framework. The Deep3DMM Comparison Platform aims at testing the hypotheses 1-3 and answering the following questions:

1. How does the normalisation of input to Deep3DMMs affect the perceptual and geometric quality of output meshes?

2. How does the standardisation of input to Deep3DMMs affect output meshes' perceptual and geometric quality?

3. Which model and representation combinations optimise the geometric and perceptual quality of the output meshes?

The overview of the Deep3DMM Comparison Platform and the strategy to test hypotheses 1-3 and answer research questions 1-3 are provided in Section 3.2. The details of the proposed platform software and the formulae to calculate inputs in different representations and the loss functions are described in Section 3.3. Section 3.4 covers the technical aspects of the models' implementation. The results from 60 experiments conducted in the Deep3DMM Comparison Platform are evaluated in Section 3.5. Evaluation described in this section proves the hypotheses 1-3 and gives detailed answers to questions 1-3.

3.2 Method Overview

This section outlines the methodology, which tests the hypotheses 1-3 from Section 3.1.1 and answers the research questions 1-3 from Section 3.1.2.

Deep 3D morphable models (Deep3DMMs) learn hierarchical, multi-scale representations of 3D meshes analogically to learning hierarchical representations of structured 2D and 3D grids, such as images and voxels. In recent years, multiple approaches have been proposed to generalise the building blocks of deep neural networks to non-Euclidean domains. The following section describes the design of the proposed Deep3DMM Comparison Platform software, which combines the building blocks of 3D morphable models to allow fair comparison between different approaches on various datasets. Moreover, the resulting software is the basis of further developments at Humain Limited. The methodology behind the execution of 60 different experimental configurations is explained in Section 3.3.7. Software design details are explained in Appendix A.

3.3 Deep3DMM Comparison Platform

The proposed Deep3DMM Comparison Platform focuses on evaluating various Deep 3D morphable models. Specifically, the FeaStNet [129], Neural 3DMM [14], SpiralNet++ [46], Mesh Autoencoder [151] and LSA-3DMM [43] are compared in this work. This section gives insight into the compared models, datasets, components of the comparison platform and formulae used to process the data, calculate loss and evaluate the results.

3.3.1 Compared models

All the compared methods are convolutional graph autoencoders or convolutional variational graph autoencoders which perform 3D mesh representation learning in spatial domain (see Chapter 2). These methods are chosen based on the following selection process. Firstly, methods that rely on 2D domain projections, such as UV mapping or image-based convolutions, are excluded due to parametrisation artifacts. In contrast, graph convolutions learn directly on the graph structure, preserving the intrinsic connectivity of the mesh. Secondly, methods which support vertex-based signal representations are preferred. This property aligns well with common mesh representations, where representations such as Euclidean coordinates or differential representation (DR) are defined on graph nodes. Finally, convolutional operators in the spectral domain are excludes as they are inherently isotropic, which limits their ability to learn anisotropic mesh features, especially in the 3D shape reconstruction task. Therefore, five representative graph convolutional autoencoders operating in the spatial domain are selected.

The models selected for comparisons differ in at least one of three main areas: graph simplification algorithm, graph convolution approach or features aggregation method.

Neural 3DMM [14] proposes spiral graph convolution, which locally orders the vertices in a spiral and keeps this order fixed. The reference vertex on the template mesh is manually selected and the spiral sequences follow the shortest geodesic distance from that vertex. To maintain a consistent spiral size, zero-padding is applied to vertices with shorter spiral lengths than the average. Pooling is defined based on quadric mesh simplification algorithm [44], which iteratively contracts edges to simplify meshes and minimise quadric error.

FeaStNet [129] introduces a novel graph convolution operator which dynamically learns the correspondence between filter weights and neighbouring nodes. FeaStNet aggregates features from neighbouring vertices based on trainable weight functions. Initially, this method was proposed for shaper correspondence task. Therefore, in this comparison, FeaStNet's convolution operation is used within the shape reconstruction framework used by Neural 3DMM method.

SpiralNet++ [46] improves upon Neural 3DMM's formulation of spiral graph convolution. The method does not limit the spiral length and therefore avoids zero-padding. Additionally, a given vertex always follows the same spiral sequence across meshes, regardless of the initial direction. This eliminates the need for manually selecting a reference vertex. SpiralNet++ performs pooling and de-pooling based on quadric mesh simplification.

Unlike FeaStNet, Mesh Autoencoder [151] and LSA-3DMM [43] design local convolutions which are not based on input features and can be shared across samples with the same topology. LSA-3DMM and Mesh Autoencoder use local structure-aware anisotropic convolutional operations. Adaptive weighting matrices are learned for each vertex based on the local neighbourhood connectivity. LSA-3DMM uses quadric mesh simplification for pooling, while Mesh Autoencoder introduces simple graph simplification based on nodes connectivity only. The method operates on non-manifold graphs and extends its graph convolution operation to transpose convolutions and residual layers. Therefore, it is the only method which allows for a fully-convolutional variational graph autoencoder. Transpose convolutions, residual layers and simple sampling are covered in detail in the following sections and in Appendix A.

3.3.2 Mesh sampling

In the context of Deep3DMMs, mesh sampling aims to create a mesh hierarchy with different levels of detail, where a mapping exists between the vertices at lower resolution and the vertices at higher resolution. Subsequently, such a hierarchy can be used in convolutional and pooling operations to capture a mesh's global and local features.

While the downsampling of 2D images is uncomplicated due to the regularity of grids, the irregular topology of graphs poses a challenge. It is addressed by many approaches to sampling meshes, outlined in Chapter 2. As downsampling is a lossy operation, upsampling transformation is usually built together with downsampling transformation. The result is a sequence of graphs $\{\mathscr{M}_0, ..., \mathscr{M}_N\}$, where N is the number of downsampled graphs, and $\mathscr{M}_0$ is the original, sampled mesh. The correspondence between the graph $\mathscr{M}_n$ and the i^{th} node of its sampled version $\mathscr{M}_{n+1}$ is defined as a local region $\mathscr{N}(i)$ in $\mathscr{M}_n$.

3.3.3 Graph convolution

Convolutions on graphs have the same purpose as traditional convolutions on 2D and 3D grids. They regularise the neural network by abstracting an input into a feature map. Similarly to downsampling, the generalisation of convolutional operations to meshes is non-trivial. Therefore, many approaches have been proposed. Chapter 2 reviews convolutional

operations on non-Euclidean domains. In general, a convolution operation calculates output feature $\mathbf{y}_i \in \mathbb{R}^O$ residing on the i^{th} node, such that:

$$\mathbf{y}_i = \sum_{\mathbf{x}_{i,j} \in \mathscr{N}(i)} \mathbf{W}_j^T \mathbf{x}_{i,j} + \mathbf{b} \ , \tag{3.1}$$

where $\mathbf{x}_{i,j} \in \mathbb{R}^I$ is the input feature residing on the j^{th} vertex in a local region $\mathscr{N}(i)$. Apart from mapping the input signal onto a different number of filters with different sizes, some approaches to graph convolutions, such as [151], allow for convolution stride larger than 1, which leads to downscaling of a graph.

3.3.4 Pooling and unpooling

Pooling is an operation which aggregates features residing on nodes in a local region $\mathscr{N}(i)$. The local region represents the connections between sampled graphs. Due to the uneven distribution of vertices and irregular topology, recent works proposed various aggregation functions to address these issues. More details on different feature aggregation methods can be found in Chapter 2. Unpooling is the opposite of pooling, so it distributes the signal to nodes in a local region $\mathscr{N}(i)$.

3.3.5 Residual layer

The introduction of residual layers to Deep3DMMs comes from direct inspiration from their equivalent in traditional deep learning architectures, particularly ResNet [52]. In the residual layer, the signal from the previous layer is aggregated with the signal from the current layer. In the case of deep learning on meshes, the following challenges can be identified: a) The previous layer and the current layer can have a different topology, and b) Each layer can have different size of features. While the former challenge can be treated as a form of pooling, the latter challenge requires a dedicated solution. Out of the methods compared in this work, only [151] proposes residual blocks. This method introduces a learned matrix with shape $O \times I$, which maps features across all the vertices in a graph, where O is the size of the output feature, and I is the size of the input feature. With solutions to both challenges, the residual layer can be constructed by mapping the features from a previous layer to a new, compatible shape and subsequently pooling or unpooling to match the topology in the current layer. Naturally, when the dimensionality of features is compatible, the mapping operation becomes an identity function. Similarly, pooling or unpooling is redundant when the topology is compatible. Thus it should be skipped.

3.3.6 Datasets

Three datasets of human head meshes are selected for comparisons. Facsimile™ [59] is a proprietary dataset owned by the industrial partner of this research, Humain Limited. It is comprised of 202 diverse identities with a neutral head mesh and 19 expressions. The meshes share common quad topology and are triangulated as part of pre-processing. Mesh resolution is 14,921 vertices. Each mesh in Facsimile™ dataset is cleaned-up by a professional artist and it aligns with quality of meshes used in AAA games.

FaceScape [144] dataset contains triangle meshes of 938 subjects with 20 expressions each. After mesh quality review, 121 subjects are excluded and 807 subjects are used in experiments. The meshes share common topology and their resolution is 26,317 vertices. FaceScape dataset is selected for comparative experiments due to high quality and one of the largest number of subjects available publicly for research purpose at the time of conducting these experiments.

FaceWarehouse [18] dataset has 150 subjects with 20 expressions each. Triangle meshes share common topology with 11,510 vertices. The dataset is selected for comparative experiments due to its popularity in 3D computer vision research.

3.3.7 Experimental configurations

Deep3DMM Comparison Platform is designed to run multiple similar experiments in the common framework, where the experiments differ in using particular types of objects, preprocessing transformations, datasets or hyperparameters. Therefore, the modular approach in designing and implementing the Deep3DMM Comparison Platform is reflected in the generation of configuration files for each experiment. For this purpose, the Hydra [143] configuration management system is used. The framework allows the dynamic creation of hierarchical configuration files, which an experiment-specific configuration file can overwrite. For example, the model configuration file comprises configuration files associated with architectures, forming a natural hierarchy of configuration files.

To test the hypotheses 1-3 and answer the research questions 1-3, 60 different hierarchical configuration files are generated. The experiments associated with each configuration differ using the respective Deep3DMM method [14, 46, 129, 151], the dataset [18, 59, 144] and the input and output representation.

3.3.8 Data processing

Rigid registration transform

Rigid registration is a process of finding the transformation in Euclidean space, which aligns two sets of points such that the overall point-wise distance between these sets is minimised. Unlike in non-rigid registration, the transformations involve only the rotation $\mathbf{R}_i$ and translation $\mathbf{t}_i$. In this work, the focus is on the rigid alignment of 2 sets of points of 3D meshes, mean $\bar{\mathbf{P}}$ and $\mathbf{P_i}$, where the correspondence between the points is known. In meshes with an uneven distribution of vertices on a surface, excluding a subset of vertices from calculating the transformations can be desirable. It is not required for the datasets considered in this work. The solution to finding $\mathbf{R}_i$ and $\mathbf{t}_i$ is minimisation of the following energy $E(\mathbf{R}_i, \mathbf{t}_i)$ in the least-squares sense:

$$\begin{aligned} \bar{\mathbf{P}} &= \frac{1}{n}\sum_{i=0}^{n}\mathbf{P}_i, \\ E(\mathbf{R}_i, \mathbf{t}_i) &= \left\| \mathbf{R}_i\mathbf{P}_i + \mathbf{t}_i - \bar{\mathbf{P}} \right\|_2 . \end{aligned} \tag{3.2}$$

Rotation $\mathbf{R}_i$ can be calculated using the singular value decomposition

$$\begin{aligned} \mathbf{U}\Sigma\mathbf{V}^T &= (\bar{\mathbf{P}} - \bar{\mathbf{p}})(\mathbf{P}_i - \bar{\mathbf{p}}_i), \\ \mathbf{R}_i &= \mathbf{V}\mathbf{U}^T, \end{aligned} \tag{3.3}$$

where $\bar{\mathbf{p}}$ and $\bar{\mathbf{p}}_i$ are the centroids of $\bar{\mathbf{P}}$ and $\mathbf{P}_i$. If $\det(\mathbf{R}_i) = -1$, then $\mathbf{R}_i$ performs reflection of the points in 3D space. In such case, the elements of the third column of $\mathbf{V}$ need to be multiplied by -1. Following that, $\mathbf{t}_i$ is obtained as follows:

$$\mathbf{t}_i = \bar{\mathbf{p}}_i - \mathbf{R}_i\bar{\mathbf{p}} \tag{3.4}$$

The resulting rigid registration function is denoted as:

$$\mathsf{rigid\_reg}(\bar{\mathbf{P}}, \mathbf{P}_i) = \mathbf{R}_i\mathbf{P}_i + \mathbf{t}_i \tag{3.5}$$

In this work, the rigid registration is used in several circumstances. Firstly, it is experimentally found that rigidly registering the meshes to their mean improves the model's performance. The improved performance comes from rigid registration transformation, which limits rotational and translational variance of data input to the neural network. Apart from preprocessing, rigid registration has a second practical application. Meshes output by the neural network in the Deformation Representation (DR) or the normalised Deformation Representation (DR Norm) need to be transformed back to Euclidean coordinates, as Eu-

clidean coordinates representation is the target representation for all the facial blendshapes. As DR and DR Norm are differential representations, the information on the location of a 3D mesh in 3D space is lost in the encoding process. Therefore, to evaluate the geometric and perceptual accuracy of the results, the meshes output from the network are decoded to Euclidean coordinates representation, and then they are rigidly registered to the ground truth mesh for evaluation with L1, L2, DAME and FMPD error metrics. In the method proposed in Chapter 4, it is no longer required to rigidly register the outputs decoded from the DR Norm representation due to a mesh assembly method, which uses spectral mesh filtering.

Centring and uniform scaling transforms

Centring and uniform scaling are the transforms implemented in the Centre and Scale classes. Centring the vertex positions $\mathbf{P}_i$ is calculated as follows:

$$\mathsf{centre}(\mathbf{P}_i) = \mathbf{P}_i - \frac{\mathsf{min}(\mathbf{P}_i) + \mathsf{max}(\mathbf{P}_i)}{2} \tag{3.6}$$

In uniform scaling, all vertex positions $\mathbf{P}_i$ are scaled by the scaling factor s, such that all the vertices end up in the box of edge length $= 2$, centred at the origin of a 3D space. In other words, the vertex positions end up within range $[-1, 1]$.

$$\begin{aligned} s = \mathsf{max}(\{\mathsf{max}(\mathbf{P}_0), \mathsf{max}(\mathbf{P}_1), ..., \mathsf{max}(\mathbf{P}_n)\}). \\ \mathsf{uniform\_scale}(\mathbf{P}_i, s) = \frac{\mathbf{P}_i}{s} \end{aligned} \tag{3.7}$$

It is experimentally found that centring and subsequently scaling the meshes input to deep 3D morphable models improves the geometric and perceptual quality of the meshes reconstructed by the model.

Standardisation and normalisation transforms

Usually, machine learning algorithms relying on gradient descent perform poorly when input feature channels have different scales. The inputs are often standardised or normalised to overcome this issue. Standardisation and normalisation are the transforms implemented in the Standardise and the Normalise classes. The inverse of these transforms is implemented in the Destandardise and Denormalise classes, respectively.

Normalisation adjusts the channels of the input to a common scale. This involves shifting and scaling the values so the features are in a certain range, usually $[0, 1]$ or $[-1, 1]$. The choice of an activation function should inform the choice of the appropriate range to which the inputs are normalised. For example, when using hyperbolic tangent $tanh(\cdot) \in (-1, 1)$ activation function, the inputs should be normalised to range $[-0.95, 0.95]$ or even

$[-0.9, 0.9]$ to avoid the gradient vanishing problem. In the case of Exponential Linear Unit *ELU* [24] activation function used in the methods compared in this work, inputs in range $[-1, 1]$ are sufficient.

The formula to normalise each channel of $\mathbf{F}_i$ to range $[a, b]$ is:

$$\mathsf{norm}(\mathbf{F}_i, a, b) = \frac{\mathbf{F}_i - \mathsf{min}(\mathbf{F})}{\mathsf{max}(\mathbf{F}) - \mathsf{min}(\mathbf{F})}(b-a) + a, \tag{3.8}$$

where $\mathsf{min}(\mathbf{F})$ denotes the minimum value in each channel of all samples $\mathbf{F}$, and $\mathsf{max}(\mathbf{F})$ denotes the maximum value in each channel of all samples $\mathbf{F}$.

In this work, the following simplified formula is used, which scales and shifts the inputs to range $[-a, a]$:

$$\mathsf{norm}(\mathbf{F}_i, a) = \frac{\mathbf{F}_i - \mathsf{min}(\mathbf{F})}{\mathsf{max}(\mathbf{F}) - \mathsf{min}(\mathbf{F})} 2a + \mathsf{min}(\mathbf{F}) \tag{3.9}$$

To denormalise the input, the following formula is used:

$$\mathsf{denorm}(\mathbf{F}_i, a) = \frac{(\mathbf{F}_i + a)(\mathsf{max}(\mathbf{F}) - \mathsf{min}(\mathbf{F}))}{2a + \mathsf{min}(\mathbf{F})} \tag{3.10}$$

Standardisation does not bind the values to a predefined range, unlike normalisation. Instead, the values of each channel are centred around the mean and scaled to have a unit standard deviation. The standardisation function is calculated as follows:

$$\begin{aligned} \sigma &= \sqrt{\frac{\sum_{i=0}^{n} (\mathbf{F}_i - \bar{\mathbf{F}})^2}{n}}, \\ \mathsf{std}(\mathbf{F}_i) &= \frac{\mathbf{F}_i - \bar{\mathbf{F}}}{\sigma} \end{aligned} \tag{3.11}$$

The inverse to standardisation is calculated as follows:

$$\mathsf{destd}(\mathbf{F}_i) = \sigma \mathbf{F}_i + \bar{\mathbf{F}} \tag{3.12}$$

Representations formulae

The inputs to deep 3D morphable models are the results of data preprocessing, which follows the aforementioned pipeline. Input representations $\mathbf{F}_{(3.14)}$ to $\mathbf{F}_{(3.14)}$ are calculated as in Equations (3.14 - 3.19).

The centred mean shape is denoted as

$$\bar{\mathbf{P}}_{ctr} = \mathsf{centre}(\bar{\mathbf{P}}). \tag{3.13}$$

Euclidean coordinates representation $\mathbf{F}_{(3.14)}$ is the result of rigidly registering the vertex locations $\bar{\mathbf{P}}$ to centred mean shape $\bar{\mathbf{P}}_{ctr}$ and subsequently uniformly scaling the vertex locations by the scalar factor s, which is calculated as in Equation (3.7). The sequence of transformations applied to obtain Euclidean coordinates is:

$$\mathbf{F}_{(3.14)} = \mathsf{uniform\_scale}(\mathsf{rigid\_reg}(\bar{\mathbf{P}}_{ctr}, \mathbf{P}), s). \tag{3.14}$$

The standardised Euclidean coordinates (Euclidean Std) representation $\mathbf{F}_{(3.15)}$ is the standardised version of $\mathbf{F}_{(3.14)}$, such that $\mathbf{F}_{(3.15)} = \mathsf{std}(\mathbf{F}_{(3.14)})$. The standardisation is calculated following the Equation 3.11. The sequence of transformations applied to obtain standardised Euclidean coordinates is:

$$\mathbf{F}_{(3.15)} = \mathsf{std}(\mathsf{uniform\_scale}(\mathsf{rigid\_reg}(\bar{\mathbf{P}}_{ctr}, \mathbf{P}), s)) \tag{3.15}$$

The normalised Euclidean coordinates (Euclidean Norm) representation $\mathbf{F}_{(3.16)}$ is the normalised version of $\mathbf{F}_{(3.14)}$, such that $\mathbf{F}_{(3.16)} = \mathsf{norm}(\mathbf{F}_{(3.14)})$. Normalisation is performed as in Equation 3.9, with $a = 1$. During the preliminary experiments, this representation has proven inferior in comparison to other representations and therefore inputs $\mathbf{F}_{(3.16)}$ are not used in comparisons presented in Section 3.5.

$$\mathbf{F}_{(3.16)} = \mathsf{norm}(\mathsf{uniform\_scale}(\mathsf{rigid\_reg}(\bar{\mathbf{P}}_{ctr}, \mathbf{P}), s), a) \tag{3.16}$$

The deformation representation (DR) is obtained from $\mathbf{F}_{(3.14)}$. The transformation function $\mathsf{dr}(\cdot)$ is implemented following the Equations from Chapter 2. The deformation gradient, calculated in the $\mathsf{dr}(\cdot)$ function, represents the displacement between the centred mean shape $\bar{\mathbf{P}}_{ctr}$ and the Euclidean coordinates $\mathbf{F}_{(3.14)}$. In other words, $\bar{\mathbf{P}}_{ctr}$ is used as the template. The sequence of transformations applied to obtain the deformation representation (DR) is:

$$\mathbf{F}_{(3.17)} = \mathsf{dr}(\bar{\mathbf{P}}_{ctr}, \mathsf{uniform\_scale}(\mathsf{rigid\_reg}(\bar{\mathbf{P}}_{ctr}, \mathbf{P}), s)) \tag{3.17}$$

The normalised deformation representation (DR Norm) is the normalised version of $\mathbf{F}_{(3.17)}$., such that $\mathbf{F}_{(3.18)} = \mathsf{norm}(\mathbf{F}_{(3.17)})$. Normalisation is performed as in Equation 3.9, with $a = 1$. The sequence of transformations applied to obtain the normalised deformation representation (DR Norm) is:

$$\mathbf{F}_{(3.18)} = \mathsf{norm}(\mathsf{dr}(\bar{\mathbf{P}}_{ctr}, \mathsf{uniform\_scale}(\mathsf{rigid\_reg}(\bar{\mathbf{P}}_{ctr}, \mathbf{P}), s)), a) \tag{3.18}$$

The standardised deformation representation (DR Std) $\mathbf{F}_{(3.19)}$ is the standardised version of $\mathbf{F}_{(3.17)}$, such that $\mathbf{F}_{(3.19)} = \mathsf{std}(\mathbf{F}_{(3.17)})$. The standardisation is calculated following

the Equation 3.11. During the preliminary experiments, this representation has proven inferior in comparison to other representations and therefore $\mathbf{F}_{(3.19)}$ is not used in comparisons presented in Section 3.5. The sequence of transformations applied to obtain the standardised deformation representation (DR Std) is:

$$\mathbf{F}_{(3.19)} = \mathsf{std}(\mathsf{dr}(\bar{\mathbf{P}}_{ctr}, \mathsf{uniform\_scale}(\mathsf{rigid\_reg}(\bar{\mathbf{P}}_{ctr}, \mathbf{P}), s))) \tag{3.19}$$

The impact of the choice of $\mathbf{F}_{(3.14)}$, $\mathbf{F}_{(3.15)}$, $\mathbf{F}_{(3.17)}$ and $\mathbf{F}_{(3.18)}$ in different Deep3DMMs on the geometric and perceptual quality of the reconstructed meshes is evaluated in Section 3.5.

3.3.9 Loss functions

Loss functions calculate the difference between the predicted values, output from a neural network, and the ground truth values. The discrepancy between the predicted and the actual values guides the optimisation process. During the training, learnable parameters of the model are updated at each iteration through backpropagation in terms of loss L. In this work, L_1 norm is used to measure this discrepancy.

Apart from using different representations input to a neural network, such as $\mathbf{F}_{(3.14)}$ (Euclidean), $\mathbf{F}_{(3.15)}$ (Euclidean Std), $\mathbf{F}_{(3.17)}$ (DR) and $\mathbf{F}_{(3.18)}$ (DR Norm), the ground truth can also be encoded in various representations. In other words, the loss function can calculate the difference between the ground truth and the predicted values in different spaces, for example, the Euclidean space or the space of differential coordinates.

Notably, the preliminary experimentation has shown that calculating the loss in a space distorted by normalisation and standardisation is suboptimal. Hence, standardised and normalised vectors predicted by the neural network are denormalised and destandardised before the loss calculation, following the Equations (3.10) and (3.12), respectively.

Moreover, it is found that the space in which the loss is calculated should match the space in which the input is represented, namely, when inputs are represented with Euclidean coordinates ($\mathbf{F}_{(3.14)}$ and $\mathbf{F}_{(3.15)}$), the loss should be calculated in the Euclidean space. Analogically, when inputs are represented with differential coordinates ($\mathbf{F}_{(3.17)}$ and $\mathbf{F}_{(3.18)}$), the loss should be calculated in the space of differential coordinates.

Based on the above discussion, the following loss functions are defined for those deep 3D morphable models which are variational autoencoders. Of the compared methods, these loss functions are used only by the Mesh Autoencoder [151]. The loss function in the variational autoencoder is a sum of two terms. The first one is L_1 norm of a difference between the predicted and the ground truth vectors. The second term is the weighted Kullback–Leibler (KL) divergence, which measures the difference between the normal

distribution $\mathcal{N}(0,1)$ and the latent distribution, where ϕ is a scalar weight. The loss terms in variational autoencoders are calculated as follows:

$$L_{(3.20)} = ||\mathbf{F}_{(3.14)} - \mathsf{denorm}(VAE(\mathbf{F}_{(3.18)}))||_1 + \phi \mathsf{KL}(\mathcal{N}(0,1)||p(E(\mathbf{Z}|\mathbf{F}_{(3.18)})), \tag{3.20}$$

$$L_{(3.21)} = ||\mathbf{F}_{(3.14)} - \mathsf{destd}(VAE(\mathbf{F}_{(3.18)}))||_1 + \phi \mathsf{KL}(\mathcal{N}(0,1)||p(E(\mathbf{Z}|\mathbf{F}_{(3.15)})), \tag{3.21}$$

$$L_{(3.22)} = ||\mathbf{F}_{(3.14)} - VAE(\mathbf{F}_{(3.14)})||_1 + \phi \mathsf{KL}(\mathcal{N}(0,1)||p(\mathbf{Z}|\mathbf{F}_{(3.14)})). \tag{3.22}$$

The remaining compared methods [14, 43, 46, 129] are the autoencoders. Their loss functions calculate the L_1 norms between the predicted and the ground truth vectors without the KL divergence regularisation term. The loss functions of the autoencoders are defined as:

$$L_{(3.23)} = ||\mathbf{F}_{(3.14)} - \mathsf{denorm}(\mathbf{F}_{(3.18)})||_1, \tag{3.23}$$

$$L_{(3.24)} = ||\mathbf{F}_{(3.14)} - \mathsf{destd}(AE(\mathbf{F}_{(3.15)})||_1, \tag{3.24}$$

$$L_{(3.25)} = ||\mathbf{F}_{(3.14)} - (AE(\mathbf{F}_{(3.14)})||_1. \tag{3.25}$$

Table 3.2 specifies the experimental configurations in which the above loss functions are used.

3.3.10 Evaluation metrics

Testing the hypotheses 1 and 3 requires measuring the geometric quality of the meshes reconstructed by a deep 3D morphable model. The geometric quality of a reconstructed mesh is defined in Chapter 1 as the L_1 or L_2 point-wise norm measurement between the predicted 3D mesh in Euclidean space and the ground truth 3D mesh in Euclidean space. Therefore, these metrics are employed to measure the geometric quality of the meshes reconstructed in the comparative experiments in this work.

The L_1 and the L_2 norms are the most common evaluation metrics used to compare the performance of different deep 3D morphable models, including the models compared in this work. However, to the knowledge of the author, this work is the first to evaluate the deep 3D morphable models in terms of the perceptual quality, defined in Chapter 1 in

terms of the Fast Mesh Perceptual Distance (FMPD) [132] or the Dihedral Angle Mesh Error (DAME) [127].

DAME metric is selected for perceptual evaluation because it has one of the highest correlation scores with human visual system (HVS) on the compression task [25]. DAME is restricted to datasets with shared topology as it is based on the difference between oriented dihedral angles in meshes. The metric takes into account the masking effect and the visibility weighting. The last one depends on the camera view and resolution, thus this term is replaced with triangle areas, as suggested in [127]. In the calculation of DAME, the border edges are ignored, as oriented dihedral angles cannot be calculated on these edges.

FMPD perceptual metric has achieved the highest overall correlation with human visual system in [25] and therefore has been selected as the second perceptual metric in the proposed comparative framework. FMPD measures discrepancy between local and global roughness of meshes. Similarly to DAME, it also accounts for the masking effect. However, unlike DAME, the metric is capable of measuring perceptual discrepancy between meshes with different connectivity. Per-vertex local roughness is calculated on the reference mesh and the evaluated mesh. Following that, the visual masking and psychosomatic saturation effect are used to weight the local roughness. These effects reflect human visual system. Next, global roughness is calculated using normalised surface integrals of the local roughness. The final perceptual score is the difference between the surface integrals of a reference mesh and the evaluated mesh.

The hypotheses 1 and 2 require the measurement of the perceptual quality, for which DAME and FMPD metrics are calculated between the predicted 3D mesh in Euclidean space and the ground truth 3D mesh in Euclidean space. Unlike the loss terms from Section 3.3.9, the evaluation metrics are always calculated in the Euclidean space. For this reason, irrespective of the model's output representation, the output is converted to Euclidean coordinates representation. The predictions in Euclidean Std representation are destandardised using the Equation 3.12, while the predictions in Euclidean representation do not require further post-processing. In contrast, DR Norm is denormalised using Equation (3.10), and both differential representations DR and DR Norm need to be converted back to Euclidean coordinates representation. As described in Section 3.3.8, the differential coordinates do not encode the location of a mesh in 3D space. For this reason, the meshes converted from the deformation representation (DR) are rigidly registered to ground truth meshes before evaluation with geometric and perceptual metrics.

3.4 Implementation Details

All models are trained with Adam optimiser [68] with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, latent space size of 64, learning rate of 10^{-3}, learning rate decay of 0.99, batch size of 16 for Facsimile and FaceWarehouse datasets and batch size of 8 for the FaceScape dataset. ELU [24] activation functions are used. The models are trained for 450 epochs.

In experiments, the Mesh Autoencoder [151] has 5 upscaling convolutional blocks and 5 downscaling convolutional blocks. The residual rates are at 0.5. Due to the nature of subsampling method used in this approach, additional graph convolutional layer was added to bring the latent space size to 64 (8 latent vertices $\times$ 8-dimensional features). Therefore, the channel dimensions are [$|\mathbf{f}|$, 32, 64, 128, 128, 8], where $|\mathbf{f}|$ is the size of a per-vertex input feature. In Euclidean coordinates-based representations $|\mathbf{f}| = 3$, while in differential coordinates-based representations $|\mathbf{f}| = 9$. The convolutional operators have stride = 2, kernel radius = 2 and basis weights = 35.

In the case of FeaStNet, Neural3DMM, SpiralNet++ and LSA-3DMM, their encoders are built of convolutional layers, each followed by a pooling layer with a pooling factor = 4. The last layer of the encoder is fully connected, with an output size equal to a latent space size of 64. The encoder channel dimensions = [$|\mathbf{f}|$, 16, 32, 64, 128]. The decoder mirrors the encoder. Additionally, in Neural3DMM, the encoder's first two layers and the decoder's last two layers are dilated convolutions with step size = 2 and dilation ratio = 2.

3.5 Experiments and Comparisons

The Deep3DMM Comparison Platform, described in Sections 3.2 and 3.3, provides a flexible environment for experimentation. The platform's modular nature allows the comparison of various deep 3D morphable model methods, and an efficient data processing pipeline allows the generation of different representations of the input. Consequently, the comparison platform is used to compare five deep 3D morphable Model approaches [14, 46, 129, 151], using three datasets [18, 59, 144] and four different input representations ($\mathbf{F}_{(3.14)}$, $\mathbf{F}_{(3.15)}$, $\mathbf{F}_{(3.17)}$ and $\mathbf{F}_{(3.18)}$), evaluated on 2 geometric metrics (L_1 and L_2 norm) and 2 perceptual metrics (DAME [127] and FMPD [132]). This comparison setup results in 60 experiments. The reconstructions from each experiment are evaluated using the training set and the test set with 4 metrics, producing 480 records in total. These records are presented in Tables 3.3, 3.4 and 3.5.

For clarity, the compared models and the inputs to these models are outlined in Table 3.1. The derivations of these inputs are shown in Section 3.3.8. The corresponding loss functions

used in each experiment are presented in Table 3.2. The equations for each loss function can be found in Section 3.3.9.

In this section, the results from the 60 experiments are evaluated from the following perspectives:

1. Impact of Euclidean coordinates-based and differential coordinates-based representations on geometric and perceptual quality of the outputs, when accounted for different Deep3DMMs and datasets (Section 3.5.1). Conclusions from this evaluation test the hypotheses 1-3.

2. Impact of standardisation and normalisation of the inputs on geometric and perceptual quality of the outputs, when accounted for different Deep3DMMs and datasets (Section 3.5.2).

3. Impact of the compared Deep3DMMs on geometric and perceptual quality of the outputs, when accounted for different input representations and training datasets (Section 3.5.3).

Table 3.1 Grid of 20 configurations used in Deep3DMM Comparison Platform. Autoencoders ($AE(\cdot)$) or Variational Autoencoders ($VAE(\cdot)$) with different inputs are laid out in terms of the Deep3DMM method and the representation. Models in these configurations are trained with 3 datasets: Facsimile, FaceWarehouse and FaceScape, resulting in 60 experimental configurations in total.

	Euclidean	Euclidean Standardised	Deformation Representation	Deformation Representation Normalised
FeaSt	$AE_{FeaSt}(\mathbf{F}_{(3.14)})$	$AE_{FeaSt}(\mathbf{F}_{(3.15)})$	$AE_{FeaSt}(\mathbf{F}_{(3.17)})$	$AE_{FeaSt}(\mathbf{F}_{(3.18)})$
Neural 3DMM	$AE_{3DMM}(\mathbf{F}_{(3.14)})$	$AE_{3DMM}(\mathbf{F}_{(3.15)})$	$AE_{3DMM}(\mathbf{F}_{(3.17)})$	$AE_{3DMM}(\mathbf{F}_{(3.18)})$
Spiral Net++	$AE_{Spiral}(\mathbf{F}_{(3.14)})$	$AE_{Spiral}(\mathbf{F}_{(3.15)})$	$AE_{Spiral}(\mathbf{F}_{(3.17)})$	$AE_{Spiral}(\mathbf{F}_{(3.18)})$
Mesh Autoe.	$VAE_{Mesh}(\mathbf{F}_{(3.14)})$	$VAE_{Mesh}(\mathbf{F}_{(3.15)})$	$VAE_{Mesh}(\mathbf{F}_{(3.17)})$	$VAE_{Mesh}(\mathbf{F}_{(3.18)})$
LSA-3DMM	$AE_{LSA}(\mathbf{F}_{(3.14)})$	$AE_{LSA}(\mathbf{F}_{(3.15)})$	$AE_{LSA}(\mathbf{F}_{(3.17)})$	$AE_{LSA}(\mathbf{F}_{(3.18)})$

Table 3.2 A grid of loss functions used in 20 configurations used in Deep3DMM Comparison Platform. The loss functions are laid out in terms of the Deep3DMM method and the representation. Models in these configurations are trained with three datasets: Facsimile, FaceWarehouse and FaceScape, resulting in 60 experimental configurations in total.

	Euclidean	Euclidean Standardised	Deformation Representation	Deformation Representation Normalised
FeaStNet [129]	$L_{(3.25)}$	$L_{(3.24)}$	$L_{(3.25)}$	$L_{(3.23)}$
Neural 3DMM [14]	$L_{(3.25)}$	$L_{(3.24)}$	$L_{(3.25)}$	$L_{(3.23)}$
Spiral Net++ [46]	$L_{(3.25)}$	$L_{(3.24)}$	$L_{(3.25)}$	$L_{(3.23)}$
Mesh Autoenc. [151]	$L_{(3.22)}$	$L_{(3.21)}$	$L_{(3.22)}$	$L_{(3.20)}$
LSA-3DMM [43]	$L_{(3.25)}$	$L_{(3.24)}$	$L_{(3.25)}$	$L_{(3.23)}$

3.5.1 Impact of Euclidean and differential representations

Quantitative evaluation

The representations are grouped into two categories: Euclidean coordinates-based (Euclidean and Euclidean Std) and differential coordinates-based (DR and DR Norm). When comparing these categories, the best-performing representation is chosen from each category to make a comparison. This is to exclude the effects of standardisation, normalisation, or the lack of thereof.

Based on the Facsimile training dataset results presented in Table 3.3, the Euclidean coordinates-based representations outperform the differential coordinates-based representations in geometric quality. Nonetheless, the differential coordinates-based representations outperform the Euclidean coordinates-based representations in perceptual quality. The L_1 norm error is lower with the Euclidean coordinates-based representations by the factor of $2.69^{+1.16}_{-1.00}$ compared to the differential coordinates-based representations. Analogically, the L_2 norm error is, on average, lower by the factor of $1.89^{+3.07}_{-1.67}$. However, with 2 out of 5 models compared, the L_2 norm error is higher with the Euclidean coordinates-based representation. The DAME error is consistently lower with the differential coordinates-based representations compared to Euclidean coordinates-based representations by the factor of $1.91^{+0.42}_{-0.62}$. The FMPD shows a similar pattern, with differential coordinates-based

Table 3.3 Quantitative comparison of the reconstruction results on the Facsimile training and test sets output from the configurations of different Deep3DMMs (in columns) using four representations (in rows). The details of experimental configurations are shown in Tables 3.1 and 3.2. The results are evaluated with L_1 norm, L_2 norm, DAME and FMPD metrics, as described in Section 3.3.10. Discussion over these results can be found in Sections 3.5.1, 3.5.2 and 3.5.3.

Facsimile Dataset - Training						
		FeaStNet	Neural 3DMM	Spiral Net++	Mesh Autoenc.	LSA-3DMM
Euclidean	L1 Norm $\times 10^{-3}$	9.27	6.595	4.777	2.602	6.881
	L2 Norm $\times 10^{-5}$	71.102	28.908	13.498	47.742	29.272
	FMPD $\times 10^{-2}$	100	93.873	100	51.962	6.433
	DAME $\times 10^{-2}$	31.542	15.306	22.990	6.036	2.85
Euclidean Std	L1 Norm $\times 10^{-3}$	2.022	1.714	1.353	2.364	1.325
	L2 Norm $\times 10^{-5}$	38.36	23.012	13.614	42.129	14.394
	FMPD $\times 10^{-2}$	35.369	25.553	44.532	43.544	8.971
	DAME $\times 10^{-2}$	5.296	3.839	5.349	4.9	2.719
DR	L1 Norm $\times 10^{-3}$	6.889	3.127	5.218	4.012	3.55
	L2 Norm $\times 10^{-5}$	26.124	57.278	14.933	9.54	71.399
	FMPD $\times 10^{-2}$	20.671	18.254	10.659	6.795	18.873
	DAME $\times 10^{-2}$	2.27	2.092	2.296	2.767	2.104
DR Norm	L1 Norm $\times 10^{-3}$	16.538	13.291	11.767	4.769	3.843
	L2 Norm $\times 10^{-5}$	135.619	83.129	68.504	13.078	87.011
	FMPD $\times 10^{-2}$	49.14	48.031	45.708	12.479	6.463
	DAME $\times 10^{-2}$	7.735	7.109	6.66	3.045	2.57

Facsimile Dataset - Test						
		FeaStNet	Neural 3DMM	Spiral Net++	Mesh Autoenc.	LSA-3DMM
Euclidean	L1 Norm $\times 10^{-3}$	9.45	6.983	6.004	8.320	7.194
	L2 Norm $\times 10^{-5}$	71.84	30.832	19.428	39.529	30.315
	FMPD $\times 10^{-2}$	100	94.536	100	48.925	6.862
	DAME $\times 10^{-2}$	31.813	15.414	22.875	5.814	2.806
Euclidean Std	L1 Norm $\times 10^{-3}$	9.07	5.954	6.38	5.782	6.367
	L2 Norm $\times 10^{-5}$	49.656	20.974	24.434	19.54	24.894
	FMPD $\times 10^{-2}$	34.374	23.965	39.42	30.846	8.396
	DAME $\times 10^{-2}$	5.354	3.807	4.869	3.885	2.939
DR	L1 Norm $\times 10^{-3}$	9.525	13.275	12.465	12.049	15.634
	L2 Norm $\times 10^{-5}$	50.34	107.757	84.487	84.892	129.809
	FMPD $\times 10^{-2}$	19.875	17.118	12.445	2.653	17.603
	DAME $\times 10^{-2}$	2.228	2.456	2.278	3.065	2.393
DR Norm	L1 Norm $\times 10^{-3}$	17.507	17.406	12.791	9.290	13.825
	L2 Norm $\times 10^{-5}$	158.564	158.41	85.656	49.449	102.034
	FMPD $\times 10^{-2}$	49.784	48.68	45.26	3.609	4.603
	DAME $\times 10^{-2}$	7.734	7.309	6.627	3	3.167

Table 3.4 Quantitative comparison of the reconstruction results on the FaceWarehouse training and test sets output from the configurations of different Deep3DMMs (in columns) using 4 representations (in rows). The details of experimental configurations are shown in Tables 3.1 and 3.2. The results are evaluated with L_1 norm, L_2 norm, DAME and FMPD metrics, as described in Section 3.3.10. Discussion over these results can be found in Sections 3.5.1, 3.5.2 and 3.5.3.

FaceWarehouse Dataset - Training						
		FeaStNet	Neural 3DMM	Spiral Net++	Mesh Autoenc.	LSA-3DMM
Euclidean	L1 Norm $\times 10^{-3}$	11.552	7.202	6.267	2.541	7.275
	L2 Norm $\times 10^{-5}$	153.49	30.02	23.225	4.403	29.636
	FMPD $\times 10^{-2}$	100	100	100	47.637	28.462
	DAME $\times 10^{-2}$	49.7	28.935	44.265	5.266	2.943
Euclidean Std	L1 Norm $\times 10^{-3}$	1.92	1.578	1.212	1.116	1.026
	L2 Norm $\times 10^{-5}$	3.193	1.902	1.041	1.091	0.896
	FMPD $\times 10^{-2}$	45.656	38.517	50.578	32.636	16.847
	DAME $\times 10^{-2}$	6.81	4.48	6.062	3.23	1.767
DR	L1 Norm $\times 10^{-3}$	6.867	7.762	4.088	2.57	4.472
	L2 Norm $\times 10^{-5}$	25.609	31.06	9.575	3.326	10.881
	FMPD $\times 10^{-2}$	1.580	1.549	0.971	3.833	2.073
	DAME $\times 10^{-2}$	0.904	0.827	0.866	1.171	0.810
DR Norm	L1 Norm $\times 10^{-3}$	9.131	7.616	6.541	2.208	3.489
	L2 Norm $\times 10^{-5}$	41.392	27.944	20.778	2.467	6.562
	FMPD $\times 10^{-2}$	10.292	10.75	10.813	3.668	1.101
	DAME $\times 10^{-2}$	2.277	2.233	2.161	1.137	1.097

FaceWarehouse Dataset - Test						
		FeaStNet	Neural 3DMM	Spiral Net++	Mesh Autoenc.	LSA-3DMM
Euclidean	L1 Norm $\times 10^{-3}$	12.177	8.043	6.996	5.332	8.295
	L2 Norm $\times 10^{-5}$	161.514	36.895	27.695	15.503	37.719
	FMPD $\times 10^{-2}$	100	100	100	48.566	26.868
	DAME $\times 10^{-2}$	49.314	29.068	44.252	5.499	2.687
Euclidean Std	L1 Norm $\times 10^{-3}$	8.169	4.017	4.694	5.337	4.472
	L2 Norm $\times 10^{-5}$	37.138	9.072	12.218	15.337	11.919
	FMPD $\times 10^{-2}$	43.419	37.621	47.91	24.573	17.532
	DAME $\times 10^{-2}$	6.299	4.458	5.633	2.528	1.89
DR	L1 Norm $\times 10^{-3}$	7.806	10.569	11.090	10.023	6.764
	L2 Norm $\times 10^{-5}$	32.325	58.098	64.124	50.386	23.561
	FMPD $\times 10^{-2}$	1.577	1.629	1.123	3.096	2.189
	DAME $\times 10^{-2}$	0.867	0.81	0.857	1.19	0.8
DR Norm	L1 Norm $\times 10^{-3}$	10.171	8.474	7.890	7.424	6.787
	L2 Norm $\times 10^{-5}$	52.726	36.706	31.728	28.66	24.309
	FMPD $\times 10^{-2}$	10.272	10.536	10.173	3.208	11.962
	DAME $\times 10^{-2}$	2.24	2.209	2.084	1.216	1.132

Table 3.5 Quantitative comparison of the reconstruction results on the FaceScape training and test sets output from the configurations of different Deep3DMMs (in columns) using 4 representations (in rows). The details of experimental configurations are shown in Tables 3.1 and 3.2. The results are evaluated with L_1 norm, L_2 norm, DAME and FMPD metrics, as described in Section 3.3.10. Discussion over these results can be found in Sections 3.5.1, 3.5.2 and 3.5.3.

FaceScape Dataset - Training						
		FeaStNet	Neural 3DMM	Spiral Net++	Mesh Autoenc.	LSA-3DMM
Euclidean	L1 Norm $\times 10^{-3}$	3.848	3.531	2.417	1.325	4.075
	L2 Norm $\times 10^{-5}$	10.335	9.886	3.799	1.586	10.178
	FMPD $\times 10^{-2}$	59.522	22.959	42.746	3.243	13.108
	DAME $\times 10^{-2}$	15.973	9.228	10.893	2.198	1.725
Euclidean Std	L1 Norm $\times 10^{-3}$	0.819	0.941	0.717	0.961	0.48
	L2 Norm $\times 10^{-5}$	0.664	0.95	0.601	1.056	0.329
	FMPD $\times 10^{-2}$	7.811	11.346	2.714	5.016	11.56
	DAME $\times 10^{-2}$	2.152	1.723	2.539	1.808	1.45
DR	L1 Norm $\times 10^{-3}$	4.997	5.022	4.458	6.948	4.67
	L2 Norm $\times 10^{-5}$	13.817	13.787	11.241	25.814	12.311
	FMPD $\times 10^{-2}$	16.787	16.246	15.757	7.927	16.612
	DAME $\times 10^{-2}$	1.439	1.447	1.458	1.666	1.402
DR Norm	L1 Norm $\times 10^{-3}$	15.337	13.592	10.139	6.068	4.473
	L2 Norm $\times 10^{-5}$	113.032	93.426	51.34	19.191	10.717
	FMPD $\times 10^{-2}$	21.267	21.057	14.353	9.633	14.719
	DAME $\times 10^{-2}$	5.505	5.298	4.406	1.638	1.632

FaceScape Dataset - Test						
		FeaStNet	Neural 3DMM	Spiral Net++	Mesh Autoenc.	LSA-3DMM
Euclidean	L1 Norm $\times 10^{-3}$	3.92	3.629	2.502	1.712	4.23
	L2 Norm $\times 10^{-5}$	10.567	10.198	3.971	2.286	10.813
	FMPD $\times 10^{-2}$	59.413	22.815	42.592	3.215	13.169
	DAME $\times 10^{-2}$	15.982	9.23	10.898	2.259	1.714
Euclidean Std	L1 Norm $\times 10^{-3}$	2.257	1.365	1.696	1.299	1.687
	L2 Norm $\times 10^{-5}$	3.845	1.54	2.178	1.381	2.547
	FMPD $\times 10^{-2}$	7.558	11.454	2.81	5.529	11.327
	DAME $\times 10^{-2}$	2.166	1.728	2.505	1.817	1.57
DR	L1 Norm $\times 10^{-3}$	6.134	6.197	4.748	7.618	5.27
	L2 Norm $\times 10^{-5}$	21.088	20.756	12.141	35.464	15.343
	FMPD $\times 10^{-2}$	17.048	16.569	16.142	9.078	16.78
	DAME $\times 10^{-2}$	1.459	1.456	1.446	1.834	1.417
DR Norm	L1 Norm $\times 10^{-3}$	15.055	13.237	9.926	5.924	4.87
	L2 Norm $\times 10^{-5}$	109.629	88.536	48.812	18.313	12.823
	FMPD $\times 10^{-2}$	20.979	20.698	14.037	10.804	15.044
	DAME $\times 10^{-2}$	5.483	5.267	4.379	1.163	1.626

representations giving the lower FMPD by the factor of $2.94^{+3.47}_{-1.94}$. Only with LSA-3DMM the FMPD remained practically the same.

The Facsimile test dataset results provide a consistent pattern, proving that Euclidean coordinates-based representations give superior geometric quality. In contrast, the differential coordinates-based representations outperform in perceptual quality. The L_1 norm error is lower with the Euclidean coordinates-based representations by the factor of $1.14^{+0.29}_{-0.20}$ compared to the differential coordinates-based representations. For the L_2 norm error, Euclidean coordinates-based representations give $3.43^{+1.71}_{-2.41}$ times lower error. The benefits of Euclidean coordinates-based representations in terms of L_1 and L_2 norms are apparent when using each of 5 compared models. On the other hand, the differential coordinates-based representations give considerably lower DAME and FMPD errors by the factors of $1.71^{+0.69}_{-0.54}$ and $1.71^{+0.69}_{-0.54}$, respectively.

Table 3.4 outlines the FaceWarehouse dataset results. Evaluation on the training set confirms the observations from analysing the Facsimile dataset results. Euclidean coordinates-based representations outperform the differential coordinates-based representations in L_1 and L_2 metrics by the respective factors of $3.43^{+1.40}_{-1.45}$ and $8.30^{+6.39}_{-6.04}$. DAME and FMPD perceptual metrics are lower with the differential coordinates-based representations compared to the Euclidean coordinates-based representations by the respective factors of $4.99^{+2.54}_{-2.81}$ and $26.01^{+26.08}_{-17.11}$.

These trends persist in the results from the test subset of the FaceWarehouse dataset. Euclidean coordinates-based representations give $1.53^{+0.58}_{-0.57}$ lower L_1 norm error, significantly outperforming the differential coordinates-based representations on all the compared models, except FeaStNet, with which it has similar L_1 norm error with both types of representations. Likewise, Euclidean coordinates-based representations outperform regarding the L_2 norm error with every model, except FeaStNet, by the factor of $2.27^{+1.77}_{-1.40}$. DAME and FMPD are consistently, significantly lower when using differential coordinates-based representations. Precisely, DAME is $4.77^{+2.50}_{-2.64}$ times lower, and FMPD is lower by the impressive factor of $21.85^{+20.82}_{-13.91}$.

The FaceScape dataset results presented in Table 3.5 paint a more complex picture. Evaluation of the training set clearly shows that Euclidean coordinates-based representations result in considerably lower L_1 norm and L_2 norm error by the factors of $6.66^{+2.66}_{-1.32}$ and $20.95^{+11.62}_{-6.44}$ respectively. Contrary to trends observed with the Facsimile and FaceWarehouse datasets, Euclidean coordinates-based representations give lower FMPD by $2.52^{+2.77}_{-1.24}$. Notably, in this case, FMPD and DAME metrics are not consistent because the differential coordinates-based representation yields lower DAME by the factor of $1.31^{+0.43}_{-0.28}$, regardless of the model used. Therefore, except for the FMPD metric, the trends on the FaceScape dataset are consistent with the previous observations. However, given minor improvements of the differential coordinates-based representations on DAME met-

ric and their significant underperformance on L_1 and L_2 metrics, standardised Euclidean coordinates representation is the preferred representation.

The results from the test set of the FaceScape dataset show that Euclidean coordinates-based representations result in $3.5^{+1.06}_{-0.78}$ times lower L_1 norm error, $8.57^{+4.91}_{-3.53}$ times lower L_2 norm error, $2.57^{+2.43}_{-1.24}$ times lower FMPD. Similarly to the training set, FMPD and DAME perceptual metrics are inconsistent, and the differential coordinates-based representations give $1.41^{+0.32}_{-0.31}$ times lower DAME.

Qualitative evaluation

The Facsimile meshes reconstructed by Deep3DMMs, which use the Euclidean coordinates-based (Euclidean and Euclidean Std) and differential coordinates-based (DR and DR Norm) representations, are visually compared against the ground truth model in Figures 3.1 and 3.2. As in the qualitative evaluation, the best-performing representation is chosen from each category to make a comparison. This is to exclude the effects of standardisation, normalisation, or the lack of thereof. This evaluation is subjective, and, as such, it is considered supplementarily to quantitative evidence.

The qualitative evaluation of the results confirms the previous observations based on the numerical metrics. On the test set, the mesh outputs from the SpiralNet++ and Neural3DMM with Euclidean Std representation, the overall shapes are closer to the ground truth compared to the mesh outputs from the SpiralNet++ and Neural3DMM with DR representation. DR representation results in noticeable volume loss over the whole head, especially in the chin area. However, the Euclidean Std representations leads to high-frequency surface perturbations, while the DR feature results in a smooth surface. Nevertheless, in the meshes output with SpiralNet++ and Neural3DMM, fine surface details are lost in reconstruction, regardless of the representation. These trends are further exaggerated on the test set. The output from the models using DR representation does not resemble the ground truth mesh. In contrast, the reconstruction with Euclidean Std representation captures its silhouette despite an abundance of high-frequency surface artefacts.

The outputs from the FeaStNet demonstrate a strong contrast between the geometric accuracy and perceptual quality of Euclidean coordinates-based and differential coordinates-based representations. On the training set, the reconstruction generated with Euclidean Std representation is more similar to ground truth regarding the volumes of individual facial features. However, the surface of the mesh is noisy and scattered with small spikes. The tip of the nose is greatly affected by high-frequency perturbations. The mesh generated with DR representation has a clean surface; however the volumes of facial features are further from the ground truth across the whole face. The difference between the impact of

the representations is even more noticeable in the test set. Regardless of representation, the generated meshes are perceptually and geometrically far from the ground truth. Despite that, the silhouette of the mesh reconstructed from the model with the Euclidean Std representation is more similar to the ground truth, except from the neck. This mesh has a multitude of small spikes scattered across its whole surface. In contrast, the mesh surface reconstructed from the model with the DR representation is clean. In this case, however, the overall silhouette does not resemble the ground truth mesh and is closer to the mean of all training samples in the Facsimile dataset.

Finally, the meshes generated by LSA-3DMM and Mesh Autoencoder follow a similar pattern. The perceptual surface quality of the outputs from LSA-3DMM is slightly higher when DR and DR Norm are used. Nevertheless, the most considerable difference in the impact of representations is in the geometric quality, where differential coordinates-based representations noticeably underperform. Moreover, despite the clean surface of meshes output from the models using differential coordinates-based representation, they are smooth and lack fine surface details. The same cannot be said about the outputs from Mesh Autoencoder. Despite a few surface artefacts, the surface details on meshes generated with the model using DR and DR Norm are close to the ground truth. Conversely to the reconstructions from the Euclidean Std representation, facial features' silhouettes and volumes are significantly different from those of the ground truth.

Summary

The comparative results from 60 experiments performed with five different models, three datasets and four representations, evaluated on two perceptual and two geometric metrics, consistently show that:

1. Using the differential coordinates-based representations results in higher perceptual quality (measured with DAME and FMPD) and lower geometric quality (measured with L_1 and L_2 norms) compared to differential coordinates-based representations. This observation proves the hypothesis 2.

2. Using the Euclidean coordinates-based representations results in higher geometric quality (measured with L_1 and L_2 norms) and lower perceptual quality (measured with DAME and FMPD) than differential coordinates-based representations. This observation proves the hypothesis 3.

Despite the above trends, exceptions exist.

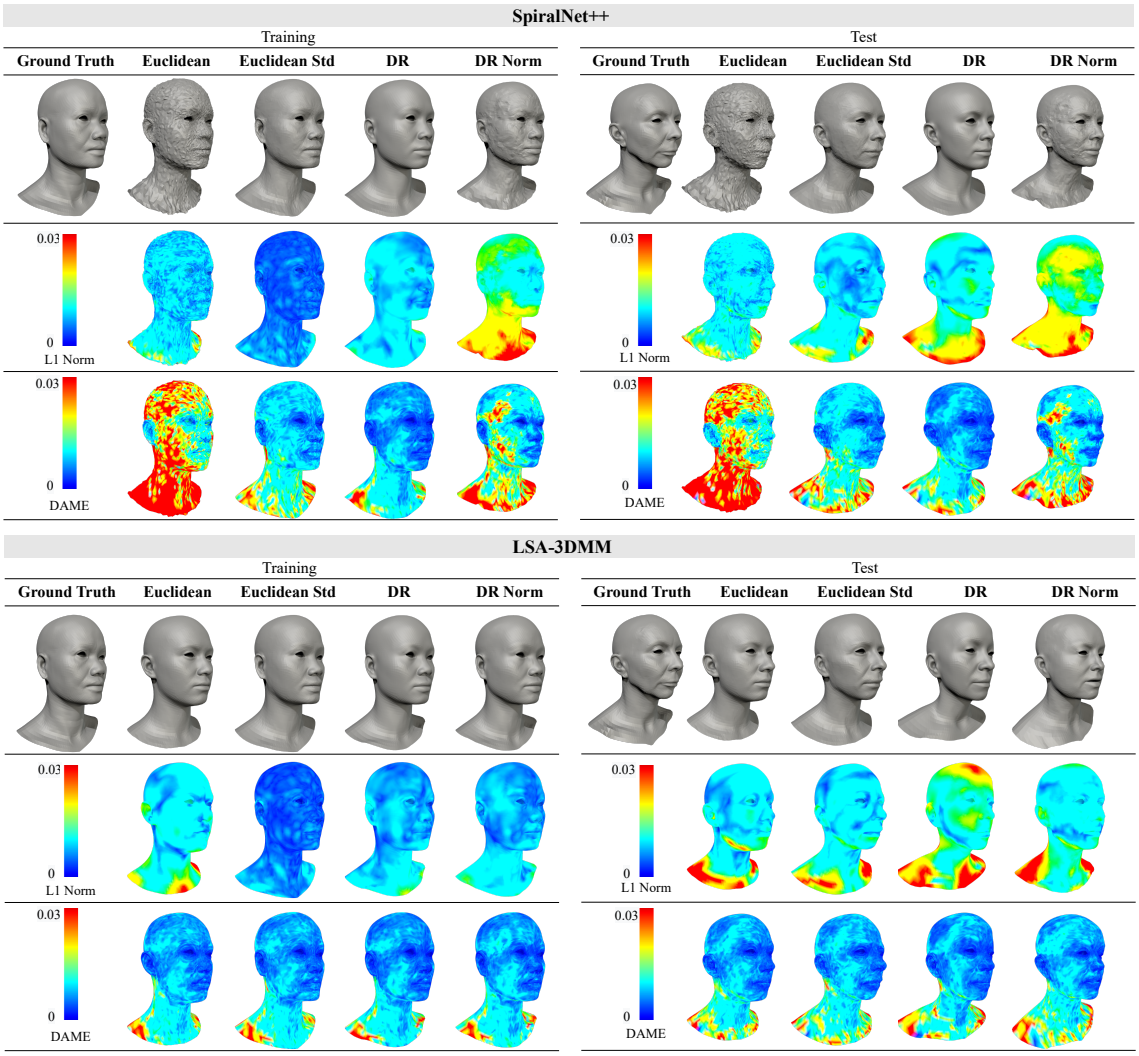


Figure 3.1 Qualitative comparison of the reconstruction results of the Facsimile training and test sets output from the SpiralNet++ (top) and LSA-3DMM (bottom) using 4 representations (in columns). The details of experimental configurations are shown in Tables 3.1 and 3.2. Per-vertex L_1 norm error and per-vertex DAME are rendered as colour. As the DAME metric aggregates error calculated on edges, a colour is assigned to a vertex by averaging error on its incident edges. The visibility weight is not applied when visualising DAME because the area occupied by vertex colour is already reflected in shading. It is recommended to zoom into the digital version to compare the surface artefacts on the generated meshes.

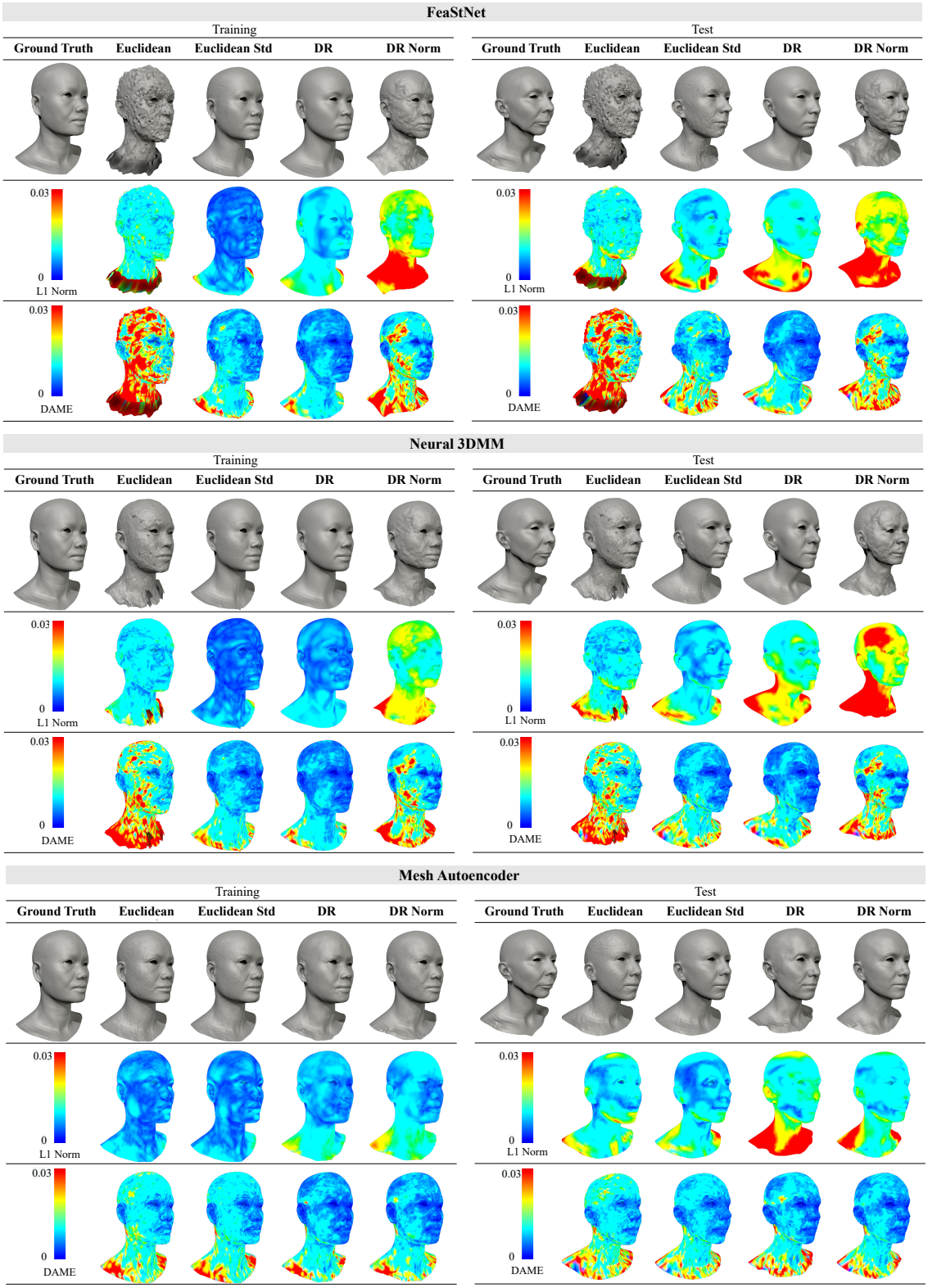


Figure 3.2 Qualitative comparison of the reconstruction results of the Facsimile training and test sets output from the FeaStNet (top), Neural3DMM (middle) and MeshAutoencoder (bottom) using 4 representations (in columns). The details of experimental configurations are shown inTables 3.1 and 3.2. Per-vertex L_1 norm error and per-vertex DAME are rendered as colour. It is recommended to zoom into the digital version to compare the surface artefacts on the generated meshes.

3.5.2 Impact of input normalisation and standardisation

Quantitative evaluation

Tables 3.3, 3.4 and 3.5 provide an insight into the effects of standardisation and normalisation. The preliminary tests have shown that $\mathbf{F}_{(3.16)}$ and $\mathbf{F}_{(3.19)}$ result in high geometric and perceptual errors and they were not included in the comparison presented in this work. The effects of standardisation are assessed by comparing the input in Euclidean coordinates representation (Euclidean) and the input in standardised Euclidean coordinates representation (Euclidean Std). The effects of normalisation are assessed by comparing the input in the deformation representation (DR) and the input in the normalised deformation representation (DR Norm).

Generally, standardisation applied to Euclidean coordinates representation improves the geometric and perceptual quality of the output meshes. On training sets, standardisation decreased the L_1 norm error by the factors of $3.65^{+1.54}_{-2.55}$, $5.02^{+2.07}_{-2.75}$ and $4.34^{+4.15}_{-2.96}$ on Facsimile, FaceWarehouse and FaceScape datasets, respectively. Standardisation results in $1.45^{+0.58}_{-0.42}$ times lower L_2 norm error on the Facsimile dataset with all the models except SpiralNet++, with which standardisation practically does not affect the L_2 norm error. On FaceWarehouse and FaceScape datasets, standardisation decreases the L_2 norm error by the respective factors of $24.66^{+23.42}_{-20.62}$ and $12.95^{+18.00}_{-11.44}$. The impact of standardisation on the perceptual metrics is also positive, regardless of the model used. It decreases DAME by $1.91^{+0.42}_{-0.62}$, $4.99^{+2.54}_{-2.81}$, $1.31^{+0.43}_{-0.28}$ on Facsimile, FaceWarehouse and FaceScape datasets, respectively. Moreover, standardisation improves FMPD by $3.30^{+2.65}_{-2.26}$, $4.87^{+2.43}_{-3.24}$, $3.89^{+3.53}_{-2.71}$, respectively.

Standardisation of the vectors in Euclidean coordinates representation on the test sets almost always improves the quality of generated meshes, with a few exceptions. Using standardised Euclidean coordinates representation decreases the L_1 norm error by $1.14^{+0.29}_{-0.20}$ on the Facsimile dataset, except from the SpiralNet++ model, with which it increases the L_1 norm error by 6%. On the other hand, standardisation decreases the L_1 norm error with the SpiralNet++ model on the FaceWarehouse dataset, while does not affect the L_1 norm error when using the Mesh Autoencoder. Nonetheless, standardisation decreases the L_1 norm error on the FaceWarehouse test set by $1.15^{+0.25}_{-0.39}$. Consistent improvements by the factor of $1.94^{+0.72}_{-0.62}$ can be observed across the results from all the models on the FaceScape dataset. The standardisation has an overall positive effect on L_2 norm error. On the Facsimile dataset, standardisation decreases the L_2 norm error by the factor of $1.48^{+0.53}_{-0.26}$. Similarly to the results on the training set, SpiralNet++ is an exception, and standardisation increases its L_2 norm error by almost 26%. On FaceWarehouse and FaceScape datasets, standardisation results in $2.97^{+1.38}_{-1.96}$ times and $3.42^{+3.20}_{1.76}$ times lower L_2 norm error, respectively. Standardisation improves the perceptual quality of the generated

meshes in most cases. It decreases FMPD by $2.36^{+1.59}_{-1.54}$ times on the Facsimile test set, by $2.11^{+0.55}_{-0.58}$ times on the FaceWarehouse test set and by $5.35^{+9.81}_{-4.77}$ times on the FaceScape test set. Exceptions are not consistent across the datasets. Namely, standardised input to LSA-3DMM on the Facsimile dataset increases FMPD by 22% and standardised input to the Mesh Autoencoder on the FaceScape test set significantly increases FMPD by 72%. This case shows the inconsistency between FMPD and DAME, as standardised input to the Mesh Autoencoder on the FaceScape test set reduces DAME by a factor of 1.11. Overall, standardisation decreases DAME by the respective factors of $3.43^{+2.51}_{-2.47}$, $5.16^{+2.70}_{-3.74}$ and $3.88^{+3.50}_{-2.79}$ on Facsimile, FaceWarehouse and FaceScape test sets.

The impact of normalisation of the deformation representation depends on the choice of a model and the dataset. The practise of normalising the DR representation prevails in research publications where deformation representation is used in deep learning. The results from the Deep3DMM Comparison Platform demonstrate that the normalisation of shapes in the deformation representation should not be a standard practice. This work shows that although normalisation can improve the quality of the outputs in some cases, it can also increase the geometric and perceptual error in many other cases, as explained below.

The comparative results from the Facsimile dataset are presented in Table 3.3. Regardless of the model, on the Facsimile training set, the L_1 norm error is consistently higher with normalised deformation representation (DR Norm), compared to the deformation representation (DR). In this case, the normalisation results in $2.24^{+2.01}_{-1.15}$ times higher L_1 norm error. A similar trend can be observed in L_2 norm error results. Regardless of the model, normalisation has an undesirable effect of increasing the L_2 norm error by the factor of $2.76^{+2.43}_{-1.55}$. Regarding the perceptual error, normalisation increases DAME and FMPD, with one exception. Namely, FMPD is 2.92 times lower with the LSA-3DMM model and normalised deformation representation. The opposite is true with all the other models compared, where FMPD increases by $2.3^{+1.99}_{-1.95}$ when normalisation is used. Moreover, normalised DR vectors perform $2.41^{+1.00}_{-1.31}$ times worse on the DAME metric across all the models, including LSA-3DMM. Therefore, it can be concluded that normalisation has undesired effects on the Facsimile training set, regardless of the model. Interestingly, the same cannot be concluded from the Facsimile test set results.

The normalisation of the DR samples from the Facsimile test set is beneficial when training the Mesh Autoencoder and the LSA-3DMM. At the same time, it has a negative impact on the outputs from FeaStNet, Neural3DMM and SpiralNet++ models. he L_1 norm error is 1.3 times lower with Mesh Autoencoder and 1.13 times lower with LSA-3DMM, while the error gets increased with the remaining three models. Similarly, normalisation has a negative impact on the L_2 norm error when using FeaStNet, Neural3DMM and SpiralNet++ models. However, it reduces the L_2 norm error by the factors of 1.94 and 1.2

with the Mesh Autoencoder and LSA-3DMM, respectively. The impact of normalisation of the Facsimile test set on the perceptual metrics (FMPD and DAME) is negative with FeaStNet, Neural3DMM and SpiralNet++ models, and the results are unclear with the two remaining models. Specifically, normalisation decreases FMPD with LSA-3DMM by 282%, but increases FMPD with Mesh Autoencoder by 36%. In contrast, normalisation decreases DAME with Mesh Autoencoder by 2%, but increases DAME with LSA-3DMM by 36%. Nonetheless, it can be concluded that, on the Facsimile test set, the overall impact of normalisation is positive with Mesh Autoencoder and LSA-3DMM, while the overall impact of normalisation is negative with FeaStNet, Neural3DMM and SpiralNet++ models.

Unlike with the Facsimile training set, there are cases when normalisation of the deformation representation is beneficial with the FaceWarehouse training set. These cases almost overlap with those from the Facsimile test set, where normalisation has an overall negative impact on FeaStNet, Neural3DMM and SpiralNet++ models, while its impact on Mesh Autoencoder and LSA-3DMM can be considered as overall positive. Normalisation decreases the L_1 norm by 2% with Neural3DMM, by 16% with Mesh Autoencoder and by 28% with LSA-3DMM. However, it increases the L_1 norm with FeaStNet and SpiralNet++ by 32% and 60%, respectively. The same trend is observed in the L_2 norm results. Normalisation decreases the L_2 norm by 11% with Neural3DMM, by 35% with Mesh Autoencoder and by 66% with LSA-3DMM, while it increases the L_2 norm with FeaStNet by 62% and SpiralNet++ by 117%. The impact on the perceptual metrics follows the trend observed in the L_1 norm error results. When normalisation is applied to DR vectors in FeaStNet, Neural3DMM and SpiralNet++ models, FMPD and DAME increase. In LSA-3DMM, DAME increases, but FMPD is 1.88 times lower. Moreover, the Mesh Autoencoder slightly benefits from normalisation, as DAME and FMPD decrease by the factors of 1.03 and 1.04, respectively.

On the FaceWarehouse test set, normalisation often improves the geometric quality of the reconstructed meshes. However, its impact is either neutral or negative on the perceptual quality. Normalising the deformation representation decreases the L_1 and L_2 norm error in Neural3DMM, SpiralNet++ and Mesh Autoencoder. The L_1 norm error is lower by the respective factors of 1.25, 1.41 and 1.35. The L_2 norm error is lower by the respective factors of 1.58, 2.02 and 1.76. Notably, normalisation has a negligible geometric impact in LSA-3DMM, whereas, in FeaStNet, it increases L_1 and L_2 norm error. Regarding perceptual quality, normalisation has a consistent, negative effect across all the models. In Mesh Autoencoder, the impact of normalisation on DAME and FMPD is insignificant. The other models are significantly affected, with FMPD higher by the factor of $5.71^{+3.35}_{-4.67}$ and DAME higher by the factor of $2.04^{+0.69}_{-1.01}$.

The results from the FaceScape training set follow the trends observed in the FaceWarehouse training set. With the normalisation, the L_1 and L_2 norm error increase in FeaStNet,

Neural3DMM and SpiralNet++ models, while normalisation improves these metrics in Mesh Autoencoder and LSA-3DMM. In these models, the L_1 norm error decreases by 15% and 4%, while the L_2 norm error decreases by 35% and 15%, respectively. FMPD and DAME metrics are not fully consistent, with FMPD indicating that normalisation improves the perceptual quality only in SpiralNet++ and LSA-3DMM. At the same time, DAME shows that normalisation marginally improves the perceptual quality only in Mesh Autoencoder. With other models, normalisation increases DAME and FMPD metrics.

The FaceScape test set results reveal the pattern from the training set. Normalisation decreases the L_1 and L_2 norm error only in Mesh Autoencoder and LSA-3DMM. The L_1 norm is 1.29 and 1.08 times lower in these models. At the same time, the L_2 norm decreases by the factors of 1.93 and 1.2, respectively. Analogically to the training set, normalisation increases the FMPD in FeaStNet, Neural3DMM and Mesh Autoencoder. However, it decreases FMPD in SpiralNet++ by 15% and in LSA-3DMM by 12%. The impact of normalisation on DAME also differs across the models. The improvements are observed only in Mesh Autoencoder, with DAME decreasing by 58%.

Qualitative evaluation

Figures 3.1 and 3.2 allow to visually compare the the effects of standardisation of Euclidean coordinates, as well as the effects of normalisation of the deformation representation (DR). Qualitative evaluation of the reconstructed FaceScape dataset confirms the conclusions from the numerical evaluation. Qualitative evaluation is more subjective and, as such, it is considered supplementary to quantitative evidence.

Compared to standardised Euclidean coordinates (Euclidean Std), the use of Euclidean coordinates has a detrimental effect on the perceptual and geometric quality of the results. In SpiralNet++, FeaStNet, Mesh Autoencoder and Neural3DMM, lack of standardisation results in severe surface artefacts, such as noise and long spikes. In LSA-3DMM, the surface is smooth and lacks details. In all the models apart from FeaStNet, the geometric accuracy is low, and the resulting facial meshes do not resemble their ground truth counterparts. Interestingly, on the test set with the FeaStNet model, the Euclidean representation results in higher geometric accuracy than the Euclidean Std representation. However, due to severe surface noise, the meshes produced with Euclidean representation are perceptually different to ground truth. This is an extreme example which demonstrates the importance of perceptual evaluation of the output meshes.

Regarding normalisation of the deformation representation, it leads to noticeable surface artefacts in SpiralNet++, FeaStNet and Neural3DMM. It also negatively impacts the geometric accuracy, as the overall shape of the reconstructed meshes tends towards the mean shape. In Mesh Autoencoder and LSA-3DMM, normalisation improves the

silhouette of the meshes. Additionally, in Mesh Autoencoder, normalisation reduces the number of surface artefacts while preserving fine surface details.

Summary

1. Standardisation of Euclidean coordinates representation significantly positively impacts the geometric and perceptual quality of meshes output by the deep 3D morphable model. There are few exceptions to this trend, which are explained in detail in this section. This answers the research question 2.

2. Although normalisation of the deformation representation is a common practice in deep 3D morphable models, in most cases, it has a negative impact on FeaStNet, Neural3DMM and SpiralNet++. The exceptions to this observation are detailed in this section. In Mesh Autoencoder and LSA-3DMM, normalisation can improve the output meshes' geometric or perceptual quality, depending on the dataset. This answers the research question 1.

3.5.3 Impact of different deep 3D morphable models

Quantitative evaluation

This work compares 5 deep 3D morphable models: FeaStNet, Neural3DMM, SpiralNet++, Mesh Autoencoder and LSA-3DMM. Due to the multi-objective nature of these models' assessment, it is impossible to select the best model unequivocally. Geometric quality, measured with L_1 and L_2 norms, and perceptual quality, measured with DAME and FMPD, are two different objectives. Prioritisation of these objectives depends on the practical application of Deep3DMMs. Figure 3.3 shows the scatter plots of different models using different representations in terms of two metrics, L_1 norm and DAME. These metrics represent the two objectives, and the models are located within a 2D space in respect of these objectives. This visualisation allows finding a Pareto-front of optimal solutions and selecting the solutions that minimise both objectives simultaneously.

Unfortunately, the combination of a model and representation, which achieves the lowest geometric error, tends to have higher perceptual error than many other combinations. Analogically, the combination that achieves the lowest perceptual error tends to have significantly higher geometric error than other combinations. This is partly due to properties of representations described in hypotheses 2 and 3, which are proven in Section 3.5.1. Another reason is the different ability of each model to represent certain representations of data, either Euclidean coordinates or differential surface properties.

On the Facsimile training set, SpiralNet++ with Euclidean Std vector has the lowest L_1 norm error of 1.353. Therefore, if geometric accuracy was the only objective, this

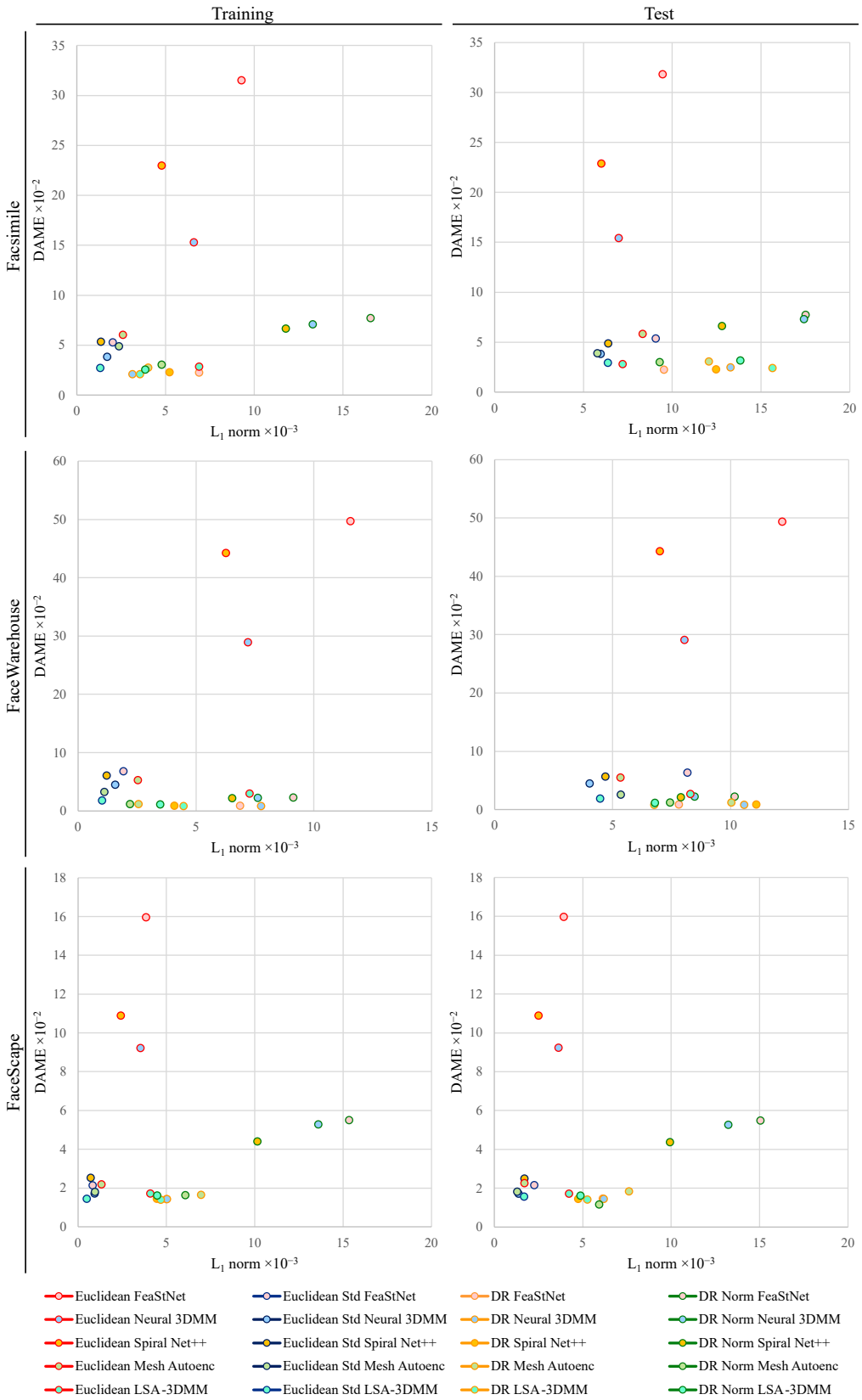


Figure 3.3 The results from the comparison of 5 Deep3DMMs configured with 4 representations plot against the L_1 norm error and the perceptual DAME metric. This visualisation allows one to simultaneously assess the models' performance in terms of both objectives. The configurations using the same model share the same fill colour, while those using the same representation share the same border colour. The discussion over these results can be found in Section 3.5.3.

combination would be the most optimal choice to reconstruct training data. However, SpiralNet++ with Euclidean Std vector achieves a high DAME, which is higher than 12 of 20 compared combinations. If perceptual quality was the sole objective, Neural3DMM with DR would be the combination of choice, as it results in a DAME of 2.092. Nonetheless, this combination with the lowest DAME has a price measured in geometric quality. Namely, its L_1 norm error is over 2.3 times higher than that of SpiralNet++ with the Euclidean Std. Based on the scatter plot in Figure 3.3, it can be deduced that the combinations which minimise both of the objectives simultaneously are LSA-3DMM with Euclidean Std (slightly in favour of geometric quality) and Neural3DMM with DR (slightly in favour of perceptual quality).

On the Facsimile test set, Mesh Autoencoder with Euclidean Std results in the lowest L_1 norm error of 6.38, compared to any other combination. Nonetheless, 10 other combinations outperform it in terms of the DAME metric. The combination which minimises DAME is FeaStNet with the DR, while FMPD is minimised by the Mesh Autoencoder with DR. Both of these combinations perform poorly on L_1 norm error. Considering both perceptual and geometric quality objectives, Mesh Autoencoder with Euclidean Std and LSA-3DMM with Euclidean Std provide a fair balance between both objectives. It is observed that although shapes in differential representation are superior regarding the perceptual quality of the results, they severely harm the geometric quality. Therefore, Euclidean coordinates-based representations are chosen as the ones which minimise both objectives simultaneously.

Based on the FaceWarehouse training set results, LSA-3DMM with Euclidean Std representation minimises the L_1 and L_2 norm error. However, this combination results in DAME higher than 7 other compared combinations. On the other hand, LSA-3DMM with DR representation minimises DAME for the price of geometric quality, where 10 other compared metrics have better performance on L_1 norm metric. When both objectives are considered, Mesh Autoencoder with DR Norm representation provides the best overall balance. When geometric quality is more favoured, LSA-3DMM with Euclidean Std vector can be a good choice, and in favour of perceptual quality, LSA-3DMM with DR representation is a reasonable choice.

On the FaceWarehouse test set, Neural3DMM with Euclidean Std representation has the lowest L_1 and L_2 norm error. Importantly, this combination also has one of the highest DAME and FMPD errors. Therefore, when both objectives are considered, there are better overall solutions than this. Regarding DAME, LSA-3DMM with DR has the lowest DAME of all combinations considered. Although 5 other combinations outperform it on L_1 norm, this combination has a good overall performance. Thus, it can be included within a set of solutions which minimise both objectives, together with LSA-3DMM with Euclidean Std representation.

Finally, the results from the FaceScape dataset are assessed. On the training set, L_1 and L_2 norm error is minimised by the combination of LSA-3DMM with Euclidean Std, with relatively low DAME error. LSA-3DMM with DR has the lowest DAME, for the price of L_1 norm error higher than 9 of 20 combinations compared. Overall, LSA-3DMM with Euclidean Std provides the best balance between perceptual and geometric quality objectives.

The situation is different on the FaceScape test set. The Mesh Autoencoder gives the best results when considering perceptual and geometric quality separately. Specifically, combining the Mesh Autoencoder with the Euclidean Std minimises the L_1 and L_2 norm error for the price of DAME higher than 8 other combinations compared. On the other hand, Mesh Autoencoder with DR Norm results in the lowest DAME for the sacrifice of geometric quality. Overall, Mesh Autoencoder with Euclidean Std optimises both objectives with a slight favour towards the geometric quality, LSA-3DMM with Euclidean Std has slightly lower DAME for the price of L_1 norm. Finally, Mesh Autoencoder with DR Norm still optimises both objectives, with favouring the perceptual quality. These three combinations can be considered a good overall choice, with an advantage towards either perceptual or geometric quality.

Qualitative evaluation

Figures 3.4 and 3.5 present the rendered meshes output from each model coupled with a representation which minimises DAME, as well as coupled with a representation which minimises L_1 norm error. Results presented in these figures allow to visually assess the strengths and weaknesses of each of the five compared models regarding the geometric accuracy and perceptual quality. Due to its subjective nature, this qualitative evaluation is only supplementary to quantitative evidence.

On the Facsimile test set, every model achieves the lowest DAME when using the DR representation, except for the Mesh Autoencoder, where DR Norm results in the lowest DAME. There is a significant variance of silhouettes across the outputs from the models. The meshes output from SpiralNet++ and FeaStNet are more similar to mean shape rather than the ground truth. The silhouettes output from LSA-3DMM, Mesh Autoencoder and Neural3DMM are much more distinct. When looking at surface details, Mesh Autoencoder outperforms other models. Despite minor surface artefacts, it preserves many fine details which are present in the ground truth. In contrast, other models produce a smooth surface, which lacks these distinct, refined features.

Among the meshes which achieve the lowest L_1 error on the test set, the output from the SpiralNet++ draws attention. The surface of the mesh is noisy, rendering the whole mesh unrecognisable from the ground truth. Notably, the surface produced by the SpiralNet++

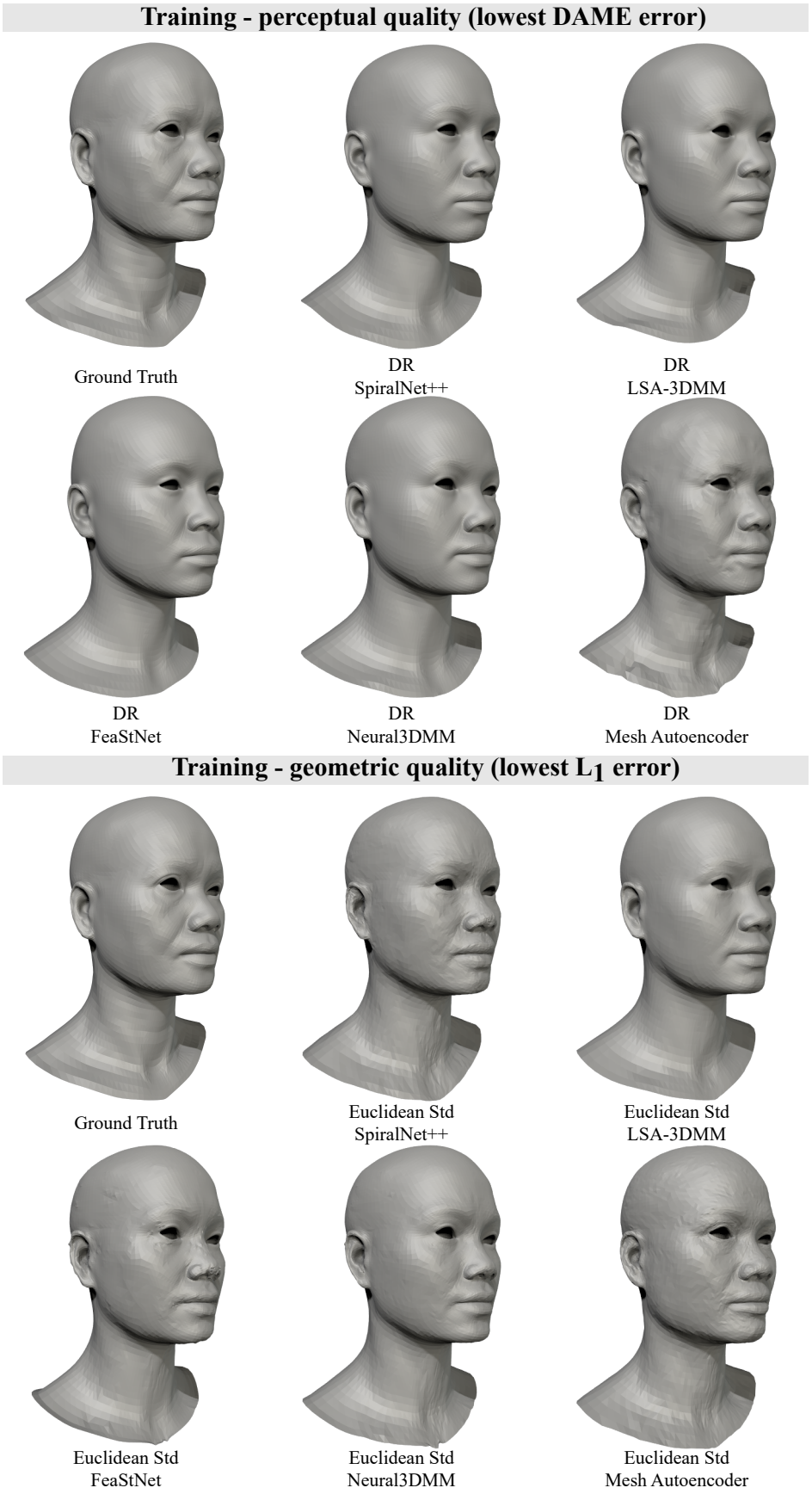


Figure 3.4 Qualitative comparison of the meshes from the Facsimile training set, reconstructed from each of 5 compared models using the representation, which achieved the highest perceptual quality (top) and the highest geometric quality (bottom).

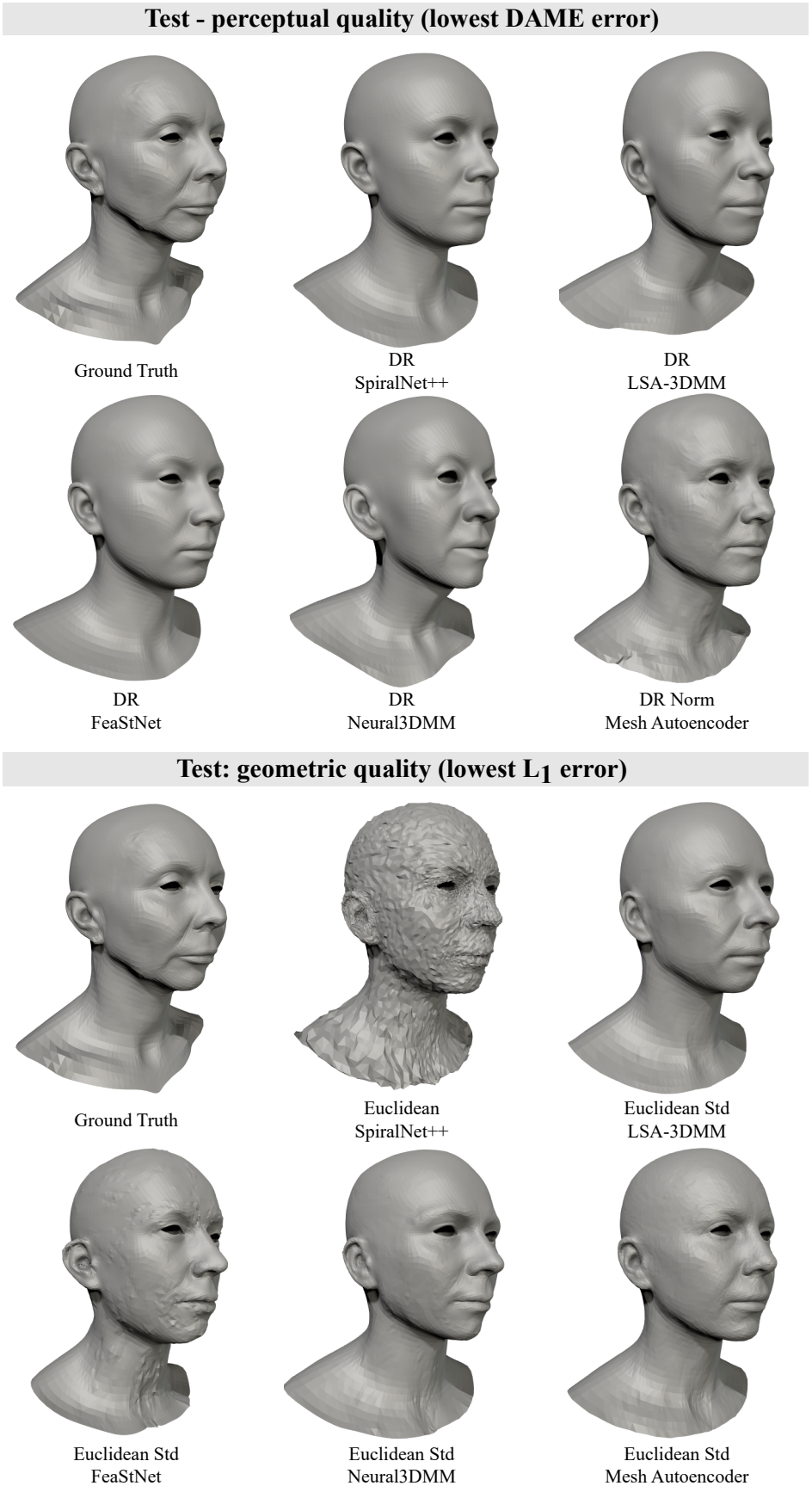


Figure 3.5 Qualitative comparison of the meshes from the Facsimile test set, reconstructed from each of 5 compared models using the representation, which achieved the highest perceptual quality (top) and the highest geometric quality (bottom).

with the Euclidean Std is much more cleaner. Neural3DMM and Mesh Autoencoder synthesise a silhouette closest to the reference.

Regarding the Facsimile training set, DR minimises DAME in all the compared models. Mesh Autoencoder captures refined features around the eyes, between eyebrows and collarbones. Despite that, it also introduces many small surface artefacts. Other models produce smoother surface.

Euclidean Std representation minimises L_1 error in all the compared models on the Facsimile set. The output from the LSA-3DMM is the most similar to ground truth regarding the surface quality and the volumes of facial features. The silhouettes produced by other models are not far from the ground truth; however, they suffer from surface perturbations, noise and spikes in case of FeaStNet.

Summary

1. The choice of an input and output representation significantly affects the geometric and perceptual quality of meshes output by different deep 3D morphable models. By altering the representation of the original method, it is possible to improve either the perceptual quality of the model's outputs (measured with DAME and FMPD) or the geometric quality of the model's outputs (measured with L_1 and L_2 norm). This proves the hypothesis 1.

2. Across three different datasets, LSA-3DMM and Mesh Autoencoder models have the best overall performance when considering the perceptual and geometric quality of the results. This answers the research question 3.

3.6 Conclusions

The proposed Deep3DMM Comparison Platform allowed the comparison of different deep 3D morphable models under a single framework. The platform's modular design provided a flexible experimentation environment in which configurations of five Deep3DMMs with four input and output representations were trained with three different datasets. The 60 experimental models were evaluated from a geometric accuracy perspective using L_1 and L_2 metrics and from a perceptual quality perspective using DAME and FMPD metrics. Quantitative and qualitative evaluation has proven hypotheses 1-3 from Section 3.1.1 and answered the research questions 1-3 posed in Section 3.1.2.

Standardisation of Euclidean coordinates representation improves the geometric and perceptual quality of meshes output by deep 3D morphable models. There exist a few exceptions to this observation. Furthermore, the findings of this work prove that the common practice of normalisation of the deformation representation is not suitable in

FeaStNet, Neural3DMM and SpiralNet++. At the same time, it can be beneficial on some datasets in Mesh Autoencoder and LSA-3DMM.

It was demonstrated that using Euclidean coordinates-based representations outperforms differential coordinates-based representations in geometric accuracy, while differential coordinates-based representations achieve better results on perceptual DAME and FMPD metrics.

The proposed use of standardised Euclidean coordinates representation improved the geometric and perceptual quality of the Mesh Autoencoder [151] method, which originally used the Euclidean coordinates. Additionally, the proposed use of the DR improved the perceptual quality of all the compared methods on most datasets. Among the proposed combinations, LSA-3DMM and Mesh Autoencoder achieved the best perceptual quality and geometric accuracy when these two objectives were considered simultaneously.

This chapter provided substantial insight into the advantages and disadvantages of representations in Deep3DMMs. Chapter 4 proposes a novel approach that trains a deep 3D morphable model that benefits from high perceptual quality of differential coordinates-based representations and high geometric accuracy of Euclidean coordinates.

This page is intentionally left blank.

CHAPTER4

DEEP SPECTRAL MESHES

4.1 Introduction

As discussed in conclusions of Chapter 3, the perceptual and geometric quality of 3D assets generated with neural models often does not meet standards to be used in industrial applications. Furthermore, the parameters exposed due to deep learning approaches, such as deep 3D morphable models, may not always be meaningful.

These issues are addressed in this chapter, in which a new method is proposed to decompose mesh into low-frequency and high-frequency displacements, and learn the latent parameters of low and high-frequency displacements. This can be achieved through introduction of spectral geometry processing to graph neural networks.

4.1.1 Inspiration by the spatial frequency theory of perception

Similarly to 3D meshes, images perceived by humans contain information at multiple spatial scales, from coarse features to fine details. According to [110], mammalian brains simultaneously create neural representations of an image at multiple frequency bandwidths. It is a consequence of neurons' receptive fields differing in size to capture these representations. At any region of the visual field, there are various sizes of receptive fields, from very small ones, which are sensitive to fine spatial details, to larger receptive fields capturing lower frequency information.

There are two observations about the spatial frequency theory of mammalian perception which are the inspiration for the method presented in this chapter:

1. Neurons in a mammalian retina are sensitive to dedicated spatial frequency bandwidths.

2. Mammalian brain simultaneously creates multiple neural representations of an image at different spatial frequency bandwidths.

In Chapter 3, it is demonstrated that certain input representations are more sensitive to coarse features of a 3D shape, while other input representations are more sensitive to fine details. Coarse features and fine details of a 3D shape are terms that require technical formulation which would expose mathematical tools to analyse them. Although not obvious at first, the context of spatial frequency theory of a visual system helps with this technical formulation. Defining coarse features of a 3D mesh as lower spatial frequency signal, and fine details of a mesh as higher spatial frequency signal, equips with a range of tools from the area of spectral mesh processing.

The findings from Chapter 3 and above technical formulation are the building blocks of a neural representation model of 3D shapes which would be closer to mammalian perception. It can be hypothesised that, similarly to neurons sensitive to different frequency bandwidths, using multiple input representations would sensitise the neural network to capture 3D shape features at different spatial resolutions. Thus, the model would leverage advantages of these input representations to faithfully reconstruct coarse features and fine details.

4.1.2 Spectral mesh decomposition in geometric deep learning

Polygon meshes [104], including quad and triangle meshes, are the most popular surface representation and are widely applied in the generation of creative content. They represent 3D surfaces with geometric vertices as absolute coordinates, making it challenging to edit overall shape displacements while preserving local surface details. In contrast, differential coordinates [39], often called the Laplacian operator or Laplacian coordinates, explicitly describe local surface and enable global shape editing while preserving local displacements [118]. Since differential coordinates are only translation-invariant and not scale- and rotation-invariant, some improvements have been made to extend differential coordinates with these additional properties. Other mesh operators [147] have also been introduced to extend the Laplacian operator. The Laplace–Beltrami operator, a generalisation of the Laplace operator, is used in this paper to represent mesh displacements.

Although mesh operators enable global shape editing while preserving local displacements, they still cannot represent displacements at different frequencies or independently edit displacements at different frequency levels. Spectral mesh processing [115] derives eigenvalues, eigenvectors, or eigenspace projections from the mesh operators and uses them to carry out desired tasks. It provides a powerful means to achieve different approximations of a 3D mesh with different frequencies. In this paper, spectral mesh processing is used to decompose mesh displacements into low- and high-frequency displacements.

Three-dimensional meshes are non-Euclidean data, unlike Euclidean data such as voxels, which have an underlying grid structure and can be treated by extending already-

existing 2D deep learning paradigms. The lack of grid structure poses a challenge when attempting to apply classical deep learning techniques to non-Euclidean data. To address this problem, geometric deep learning [15] has been developed explicitly for non-Euclidean data. Consequently, geometric deep learning is used in the proposed method to learn the latent parameters of low- and high-frequency displacements.

Parametric models, such as 3D morphable models [34], are commonly used to synthesise new meshes by altering the coefficients in a parametric space. They are widely applied due to their ability to model intrinsic properties of 3D faces. Parametric models have been used in graph neural networks to represent facial shapes. A parametric model is also used in this paper for 3D facial mesh synthesis.

As the most popular 3D structure for representing 3D models, triangle meshes are graphs. Graph neural networks are suitable for the geometric learning of triangle meshes [141]. Therefore, triangle meshes are considered in our proposed methods, and graph neural networks are used for deep learning.

By integrating the above-discussed Laplace–Beltrami operator for deformation representation, spectral mesh processing for decomposition of mesh displacements, geometric deep learning, and parametric models with graph neural networks, a new 3D facial mesh synthesis model is developed in this paper. The model exposes user parameters to control disentangled low- and high-frequency displacements, generate plausible facial shapes, and allow the user to control displacements independently at low- and high-frequency levels.

Generating plausible facial shapes and allowing controllable deformation can conflict, as plausible faces exist within a joint distribution of low- and high-frequency information. For example, wrinkled skin at higher frequencies correlates with volume loss of fat pads at lower frequencies. This problem is exacerbated when using smaller or biased datasets, which can lead to undesirable correlations. A Conditioning Factor is introduced to overcome the conflict between plausibility and user control. It is a scalar that modulates the influence of mutual conditioning of low- and high-frequency displacements.

Lower-frequency displacements encode most of the mesh volume, while higher-frequency displacements describe fine surface details. This observation is used to improve the quality of generated meshes. Low frequencies are represented with standardised Euclidean coordinates, which capture first-order mesh properties. The highest frequency displacement is represented with normalised deformation representation (DR) to improve perceptual quality. In this way, our proposed method improves the overall quality of generated meshes from geometric and perceptual perspectives, as shown in Section 4.7.

4.2 Method Overview

This section introduces the spectral decomposition of meshes in 3D shape representation learning. A method of partitioning 3D meshes to obtain multiple input vectors is described. Each partition relates to a different spatial frequency signal. Following that, graph neural network is presented. It learns latent representations for each spatial resolution. Finally, the process of assembling the final 3D meshes from the reconstructed frequency bands is described.

The proposed method considers only triangle manifold meshes that share the same topology. Non-triangular meshes need to be triangulated beforehand. Delaunay triangulation is a common method to triangulate a mesh. The original, non-triangular connectivity of meshes can be reestablished after using the proposed method.

Although the focus is on manifold meshes, this method can be generalised to non-manifold meshes, provided that they share the same topology and an alternative Laplacian matrix is formulated for Equation (4.2), such as a Laplacian for non-manifold triangle meshes in [112]. The input representations should be defined on non-manifold connectivity. Additionally, the neural network architecture from Section 4.2.2 must provide operators on non-manifold meshes. The operators that are described in Section 2.3.1 meet this requirement.

4.2.1 Spectral Partitioning and Representation

To represent different scales of detail with different input representations, a procedure is required, which consistently decomposes 3D meshes into several signals representing different spatial scale of detail. The following mesh partitioning requirements are identified. It should be possible to:

1. Assemble the signals back to the original mesh. In other words, the signals should be additive and they should add up to the signal from before the partition.

2. Ensure that the signal associated with a spatial scale of detail contains only signal at that given spatial scale of detail.

There can be considered two solutions to partitioning the 3D meshes into signals representing different spatial scale of detail. These candidate solutions are tested against the mesh partitioning requirements. The importance of satisfying these requirements is further evidenced in Section 4.2.3.

3D mesh simplification allows to obtain signals at different spatial scales of detail through coarsening of a 3D mesh [44]. It is possible to devise a mapping between different mesh resolutions and obtain a multi-resolution mesh which would satisfy the

requirement (1). Nevertheless, 3D mesh simplification operates on mesh connectivity rather than signal on mesh vertices and therefore it does not meet the mesh partitioning requirement (2). For example, it is possible to add noise associated with fine spatial detail to signal residing on a coarse connectivity and the coarsening operation does not filter out that noise to ensure that the coarse connectivity contains only coarse spatial details.

Spectral mesh partitioning allows to devise subspaces associated with different spatial frequency scales. Projection of the original signal residing on 3D mesh vertices onto each of these subspaces allows to obtain the signal associated with the given frequency scale. The resulting signals are additive and they add up to the signal from before the partition, thus satisfying the mesh partitioning requirement 1. As the procedure operates on the original signal and produces signal associated with the given frequency bandwidth by filtering out the signal outside of that bandwidth, it is possible to ensure that the signal associated with a spatial scale of detail contains only signal at that given spatial scale of detail. Therefore, the mesh partitioning requirement 2 is satisfied. For example, it is possible to add noise associated with fine spatial detail to signal associated with low frequency bandwidth. The spectral mesh partitioning operation allows to filter out the signal associated with fine spatial detail.

Following this discussion, spectral mesh partitioning is a good method of choice. The generalised spectral decomposition [147] of a Laplacian matrix $\mathbf{L}$ has a form of

$$\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^T. \tag{4.1}$$

It is found that the use of mass matrix counters the impact of irregular tessellation and, in consequence, improves the geometric quality of meshes generated with the proposed method. Therefore, the spectral decomposition of a Laplacian matrix L is formulated as

$$\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^T\mathbf{M}, \tag{4.2}$$

where $\mathbf{M}$ is the mass matrix representing Voronoi areas around vertices. Section 4.3 demonstrates the advantages of using the formulation in Equation (4.2) over using the formulation in Equation (4.1).

A mass matrix is a diagonal matrix whose entries $\mathbf{M}_{ii}$ correspond to the area around vertex i in the mesh. There are several algorithms to calculate the area around vertex i. Barycentric area and Voronoi area are common methods. The barycentric area around vertex i is one-third of a total area of triangles neighbouring the vertex i. Therefore, it is highly dependent on the connectivity of a mesh. In contrast, the Voronoi area depends solely on the position of neighbouring vertices. For this reason, the Voronoi mass matrix is chosen to be used in this method.

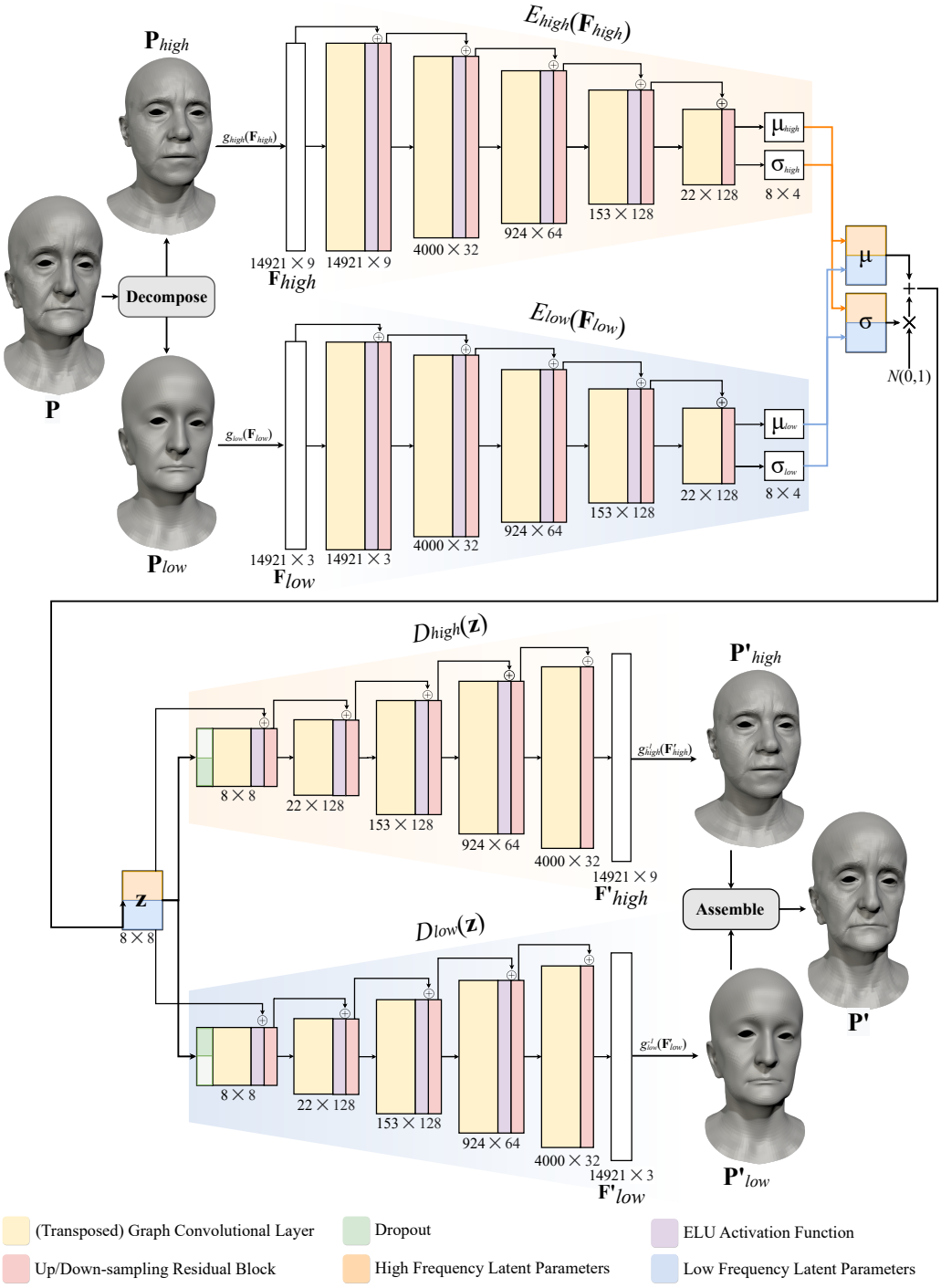


Figure 4.1 Overview of the proposed Deep Spectral Meshes graph neural network. Preprocessed meshes $\mathbf{P}$ are partitioned to two frequency bands through spectral decomposition. The resulting low and high-frequency displacements $\mathbf{P}_{low}$ and $\mathbf{P}_{high}$ are transformed to standardised Euclidean coordinates $\mathbf{F}_{low}$ and the deformation representation (DR) $\mathbf{F}_{high}$ (Section 4.4.1). Subsequently, graph encoders E_{high} and E_{low} encode the features to latent means μ_{high} and μ_{low}, and deviations σ_{high} and σ_{low}. Means and deviations are concatenated, and latent codes $\mathbf{Z}$ are sampled from the distribution $\mathcal{N}([\mu_{high} \mid \mu_{low}], [\sigma_{high} \mid \sigma_{low}])$. Graph decoders D_{high} and D_{low} reconstruct inputs from $\mathbf{Z}$ (Sections 4.4.2 - 4.4.4). Outputs $\mathbf{F}'_{high}$ and $\mathbf{F}'_{low}$ are converted back from their representations to Euclidean coordinates $\mathbf{P}'_{high}$ and $\mathbf{P}'_{low}$, which are later combined to get the final vertex positions $\mathbf{P}'$ (Section 4.4.5).

Spectral partitioning is the process of identifying n subsets of eigenvectors $\mathbf{U}$ and projecting vertex positions $\mathbf{P}$ of a mesh onto these subsets of vectors to isolate different frequency bands $\{\mathbf{P}_0, ..., \mathbf{P}_n\}, \mathbf{P} = \sum_{i=0}^{n} \mathbf{P}_i$. The choice of size and number of frequency bands is arbitrary and depends on a specific application. For $\mathbf{U}_i$ derived from Equation 4.1, the projection of $\mathbf{P}$ onto a subset of eigenvectors $\mathbf{U}_i$ is defined as

$$\mathbf{P}_i = \mathbf{U}_i \mathbf{U}_i^T \mathbf{P}. \tag{4.3}$$

For $\mathbf{U}_i$ derived from Equation 4.2, the projection of $\mathbf{P}$ onto a subset of eigenvectors $\mathbf{U}_i$ is defined as

$$\mathbf{P}_i = \mathbf{U}_i \mathbf{U}_i^T \mathbf{M}\mathbf{P}. \tag{4.4}$$

As a consequence of spectral partitioning, frequency bands $\{\mathbf{P}_0, ..., \mathbf{P}_n\}$ can be transformed into different, dedicated representations, which are more suitable for a given spectral band. A representation transform can be denoted as function $g_i(\mathbf{P}_i)$, which transforms the ith frequency band into representation $\mathbf{F}_i$. An associated inverse transform can be denoted as $g_i^{-1}(\mathbf{F}_i)$. These functions must output a per-vertex representation because $\mathbf{F}_i$ are inputs to graph neural networks which process input signal defined on vertices.

4.2.2 Neural Network

Variational graph autoencoders [151] are used to encode features of each frequency band into their respective latent distributions $\mathcal{N}(\mu_i, \sigma_i)$, where μ_i is a mean vector and σ_i is a standard deviation vector. If desired, each encoder can output a different number of parameters. The optimal size of the parametric space depends on the training dataset and requirements of a specific application. Therefore, it can be determined as part of the hyperparameter optimisation process.

The neural network consists of n graph encoders E_i with n corresponding decoders D_i. The choice of convolutional, transpose convolutional, pooling, de-pooling or other layers of encoders and decoders is arbitrary. While each encoder outputs a different latent distribution of its frequency band, the encoders take the same set of parameters as input. Specifically, mean vectors are concatenated to $\mu = [\mu_0, ..., \mu_n]$ and deviation vectors are concatenated to $\sigma = [\sigma_0, ..., \sigma_n]$. Afterwards, latent parameters $\mathbf{Z}$ are stochastically sampled from $\mathcal{N}(\mu, \sigma)$. The decoders are optimised to output reconstructed vectors $\mathbf{F}'_i$, with the objective of minimising the discrepancy between $\mathbf{F}_i$ and $\mathbf{F}'_i$.

4.2.3 Final Assembly

Before combining reconstructed vectors $\mathbf{F}'_i$, it is necessary to convert them back to Euclidean coordinates so that the reconstructed frequency band $\mathbf{P}'_i = g_i^{-1}(\mathbf{F}'_i)$. It can be observed that each decoder D_i is optimised to generate outputs which, after transformation with $g_i^{-1}(\cdot)$, are a linear combination of eigenvectors $\mathbf{U}_i$. This implies that $\mathbf{P}'_i$ can be projected onto these vectors without information loss, as the filtered-out frequencies are noise which lies outside the domain for this frequency band. Based on this observation, the final reconstructed vertex positions are obtained through the aggregation of frequency bands $\mathbf{P}'_i$ in the following way:

$$\mathbf{P}' = \sum_{i=0}^{n} \mathbf{U}_i \mathbf{U}_i^T \mathbf{M} \mathbf{P}'_i \quad . \tag{4.5}$$

4.3 Mass matrix in spectral partitioning

In this section, the impact of a mass matrix $\mathbf{M}$ in spectral mesh partitioning is discussed. The spectral partitioning with and without the mass matrix is compared. The former is based on Equations (4.1) and (4.3). The latter is based on Equations (4.2) and (4.4).

The projection of vertex positions $\mathbf{P}$ onto a subset of first k eigenvectors $\mathbf{U}_{(k)}$ corresponding to the smallest eigenvalues is considered to obtain $\mathbf{P}_{(k)}$. Vertex positions $\mathbf{P}_{(k)}$ are the results of low-pass filtering, which means that they represent low frequency information of $\mathbf{P}$. Therefore, the remaining high frequency information can be calculated as $\mathbf{P} - \mathbf{P}_{(k)}$. As parameter k increases, the lower resolution signal $\mathbf{P}_{(k)}$ is closer to original signal $\mathbf{P}$. Therefore, as parameter k increases, $||\mathbf{P} - \mathbf{P}_{(k)}||_1$ decreases.

It is desirable that as k increases, $||\mathbf{P} - \mathbf{P}_{(k)}||_1$ decreases at similar rate at all regions of the face. If that is not the case, the mesh partitioning requirement (2) from Section 4.2.1 cannot be fully satisfied, as lower spatial frequency signal would remain in $\mathbf{P} - \mathbf{P}_{(k)}$ and / or higher spatial frequency signal would remain in $\mathbf{P}_{(k)}$. The resulting imbalance would compromise the benefits of using different per-partition input representations, which are more sensitive to signal at given spatial frequency scale.

In practical applications, tessellation of 3D meshes often varies in resolution across the mesh. For example, 3D meshes of human heads tend to have more vertices in the area of a face, where finer details of facial features and their displacements need to be preserved. This is especially common around the eyes and lips, which are involved in complex facial deformations. The area of a scalp tends to have coarser tessellation, because its subject to lower frequency displacements and, in practical applications, the scalp is often hidden under hair or headgear.

Varying spatial resolution of the tessellation of a 3D mesh leads to discrepancy between the spatial frequency of a signal on the mesh vertices and the topological frequency of a signal on the mesh vertices. This is because geodesic distance between the vertices and spatial distance between these vertices is not proportional among all the vertices.

When following Equations (4.1) and (4.3), $\mathbf{P}_{(k)}$ is denoted as $\mathbf{P}_{(k)_{(4.3)}}$. Consequently, when following Equations (4.2) and (4.4), $\mathbf{P}_{(k)}$ is denoted as $\mathbf{P}_{(k)_{(4.4)}}$.

Based on this discussion, it is hypothesised that:

1. As parameter k increases, $||\mathbf{P} - \mathbf{P}_{(k)_{(4.3)}}||_1$ decreases at different rate across the vertices if the tessellation has varying spatial resolution. It decreases faster in the areas with finer tessellation (smaller areas of the faces of a mesh, such as eyes) and decreases more slowly at the areas of the coarser tessellation (larger areas of the faces of a mesh, such as neck).

2. (1) implies that as parameter k increases, $||\mathbf{P}_{(k)_{(4.3)}}||_1$ increases at different rate across the vertices if the tessellation has varying spatial resolution. It decreases more slowly in the areas with finer tessellation (smaller areas of the faces of a mesh) and it decreases faster at the areas of the coarser tessellation (larger areas of the faces of a mesh).

3. As parameter k increases, $||\mathbf{P} - \mathbf{P}_{(k)_{(4.4)}}||_1$ decreases at similar rate across all the vertices.

4. (3) implies that as parameter k increases, $||\mathbf{P}_{(k)_{(4.4)}}||_1$ increases at similar rate across all the vertices.

An experiment is designed to prove above hypotheses (1) and (3), and by implication hypotheses (2) and (4). For all $k \in \{50, 100, 150, 200, 250, 300, 350, 400, 500, 600, 700, 900, 1000, 1200, 1500, 1700, 2000\}$, $\mathbf{P}_{(k)_{(4.3)}}$ and $\mathbf{P}_{(k)_{(4.4)}}$ is computed on $m = 202$ neutral face meshes, $n = 14,921$ vertices each, from the Facsimile™ [59] dataset.

Next, the following error metrics are calculated:

$$E_{\text{ave}(k)(4.3)} = \frac{1}{m}\sum_{h=0}^{m-1}\frac{1}{n}\sum_{j=0}^{n-1}||\mathbf{p}^h_{j(k)_{(4.3)}} - \mathbf{p}^h_j||_1, \tag{4.6}$$

$$E_{\text{ave}(k)(4.4)} = \frac{1}{m}\sum_{h=0}^{m-1}\frac{1}{n}\sum_{j=0}^{n-1}||\mathbf{p}^h_{j(k)_{(4.4)}} - \mathbf{p}^h_j||_1, \tag{4.7}$$

$$\begin{aligned} E_{\max(k)(4.3)} = \frac{1}{m}\sum_{h=0}^{m-1}\max\{&||\mathbf{p}^h_{0(k)_{(4.3)}} - \mathbf{p}^h_0||_1, ||\mathbf{p}^h_{1(k)_{(4.3)}} - \mathbf{p}^h_1||_1, \\ &\ldots, ||\mathbf{p}^h_{n-1(k)_{(4.3)}} - \mathbf{p}^h_{n-1}||_1\} \end{aligned} \tag{4.8}$$

$$E_{\max(k)(4.4)} = \frac{1}{m}\sum_{h=0}^{m-1} \max\{||\mathbf{p}^h_{0(k)_{(4.4)}} - \mathbf{p}^h_0||_1, ||\mathbf{p}^h_{1(k)_{(4.4)}} - \mathbf{p}^h_1||_1, \ldots, ||\mathbf{p}^h_{n-1(k)_{(4.4)}} - \mathbf{p}^h_{n-1}||_1\} \tag{4.9}$$

4.3.1 Quantitative evaluation

Figure 4.2 compares $E_{\text{ave}(k)(4.3)}$ and $E_{\text{ave}(k)(4.4)}$ in terms of the lower frequency band size k. As k increases, on average $||\mathbf{P} - \mathbf{P}_{(k)_{(4.3)}}||_1$ decreases faster than $||\mathbf{P} - \mathbf{P}_{(k)_{(4.4)}}||_1$. It means that to achieve the same L_1 discrepancy between the lower frequency signal and the original signal, the parameter k needs to be set to a higher value when using Equations (4.2) and (4.4) comparing to using Equations (4.2) and (4.4). The requirement to use higher value of k when using the mass matrix normalisation is not computationally significant, as demonstrated in Figure 4.3. The CPU time is similar independently on the value of k. The CPU time is consistently higher by 20 ms per mesh when using the mass matrix normalisation. Nevertheless, this computational time difference is not significant, as it adds merely 4 seconds to the preprocessing time of all the neutral faces in the Facsimile™ [59] dataset.

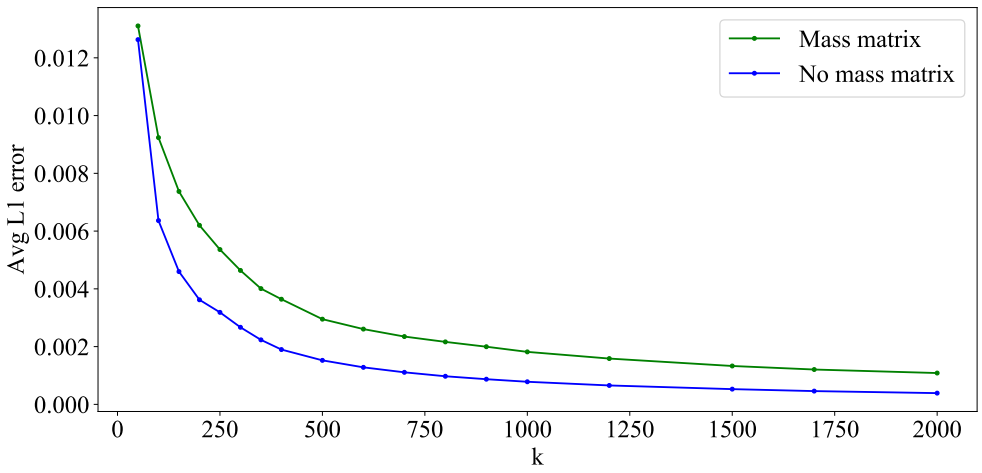


Figure 4.2 Plot of $E_{\text{ave}(k)(4.3)}$ (blue line) and $E_{\text{ave}(k)(4.4)}$ (green line) in terms of the parameter k. $E_{\text{ave}(k)(4.3)}$ consistently requires lower k to achieve the same average L_1 norm.

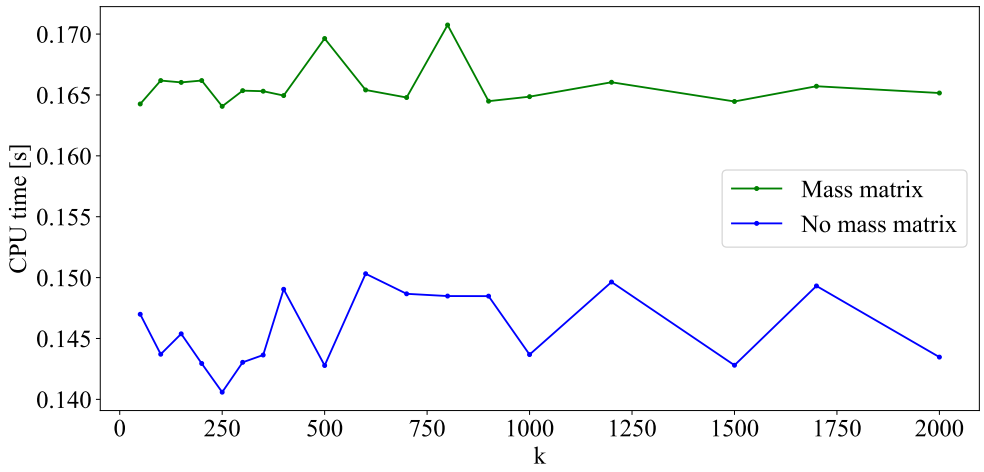


Figure 4.3 Comparison of per-mesh CPU time required to compute Equations (4.3) (blue line) and (4.4) (green line) in terms of different parameters k. It is demonstrated that CPU time is independent of value of k and that Equation (4.4) takes $\approx$ 20 ms longer to compute than Equation (4.3).

Figure 4.4 shows that despite consistently higher $E_{\text{ave}(k)(4.4)}$ than $E_{\text{ave}(k)(4.3)}$, the max error $E_{\text{max}(k)(4.4)}$ is not always higher then $E_{\text{max}(k)(4.4)}$ in terms of k. In further evaluation, it is shown that the max L_1 error is actually consistently lower with mass matrix normalisation when accounting for discrepancy between average L_1 error.

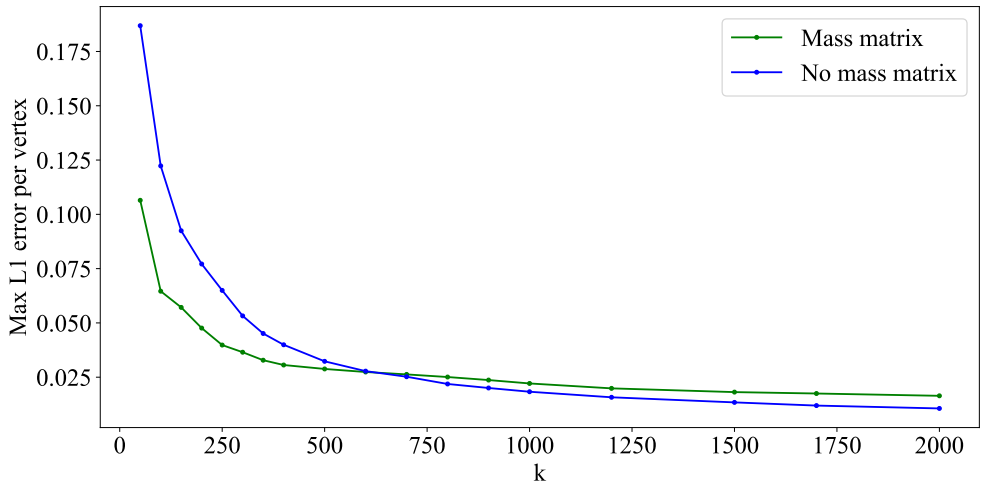


Figure 4.4 Plot of $E_{\text{max}(k)(4.3)}$ (blue line) and $E_{\text{ave}(k)(4.4)}$ (green line) in terms of the parameter k. $E_{\text{max}(k)(4.3)}$ is higher up to $k = 600$, and lower from that point.

Based on the results presented in Figure 4.2, it can be deduced that $\mathbf{P}_{(k)_{(4.3)}}$ and $\mathbf{P}_{(k)_{(4.4)}}$ are not comparable with the same value of k, because $\forall k(||\mathbf{P}-\mathbf{P}_{(k)_{(4.3)}}||_1 \not\approx ||\mathbf{P}-\mathbf{P}_{(k)_{(4.4)}}||_1)$.

To prove hypotheses (1) and (3), the distribution of L_1 error across the vertices can be evaluated. To do that, $E_{\max(k)(4.3)}$ is compared with $E_{\max(k)(4.4)}$ in terms of the L_1 error $E_{\text{ave}(k)(4.3)}$ and $E_{\text{ave}(k)(4.4)}$ respectively. Higher max error indicates less uniform distribution of L_1 error. The results are depicted in Figure 4.5. For a given L_1 error, the max error is consistently higher in the signal without mass matrix normalisation. That indicates that the distribution of the L_1 error is less uniform in $||\mathbf{P}-\mathbf{P}_{(k)_{(4.3)}}||_1$ comparing to $||\mathbf{P}-\mathbf{P}_{(k)_{(4.4)}}||_1$.

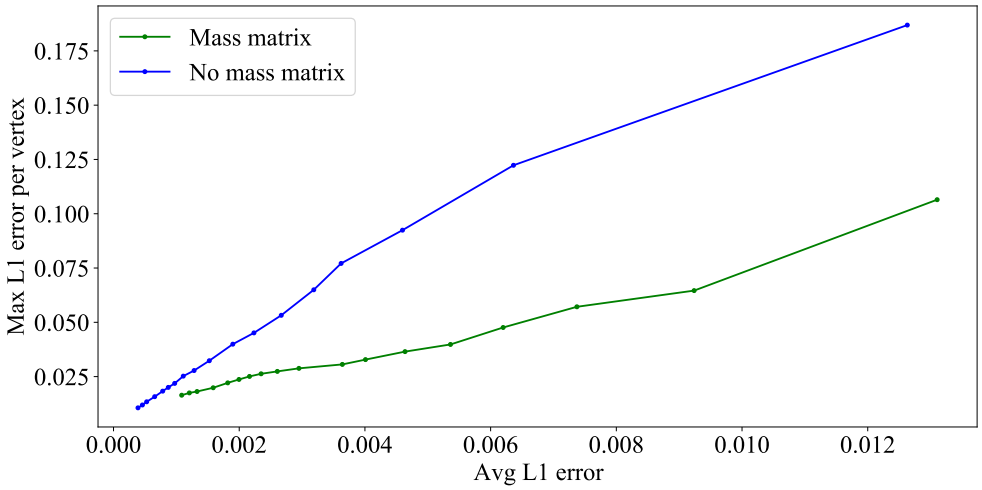


Figure 4.5 Comparison of $E_{\max(k)(4.3)}$ (blue line) and $E_{\max(k)(4.4)}$ (green line) in terms of the average L_1 error. It demonstrates that for a given average L_1 error, $E_{\max(k)(4.3)}$ is higher than $E_{\max(k)(4.4)}$.

4.3.2 Qualitative evaluation

To ensure fair comparison, a pair of hyperparameters k is selected so that the average L_1 norm between the lower frequency signal and the original signal is similar ($\pm 10^{-4}$) when using the mass matrix normalisation and without using the mass matrix normalisation. One of multitude of possible pairs is picked, $k = 500$ for $\mathbf{P}_{(k)_{(4.3)}}$ and $k = 1200$ for $\mathbf{P}_{(k)_{(4.4)}}$. This way, on average,

$$||\mathbf{P}-\mathbf{P}_{(500)_{(4.3)}}||_1 \approx ||\mathbf{P}-\mathbf{P}_{(1200)_{(4.4)}}||_1.$$

Figure 4.6 presents randomly selected meshes $\mathbf{P}_{(1200)_{(4.4)}}$ and their corresponding meshes $\mathbf{P}_{(500)_{(4.3)}}$. The spatial frequency of the signal varies significantly across the

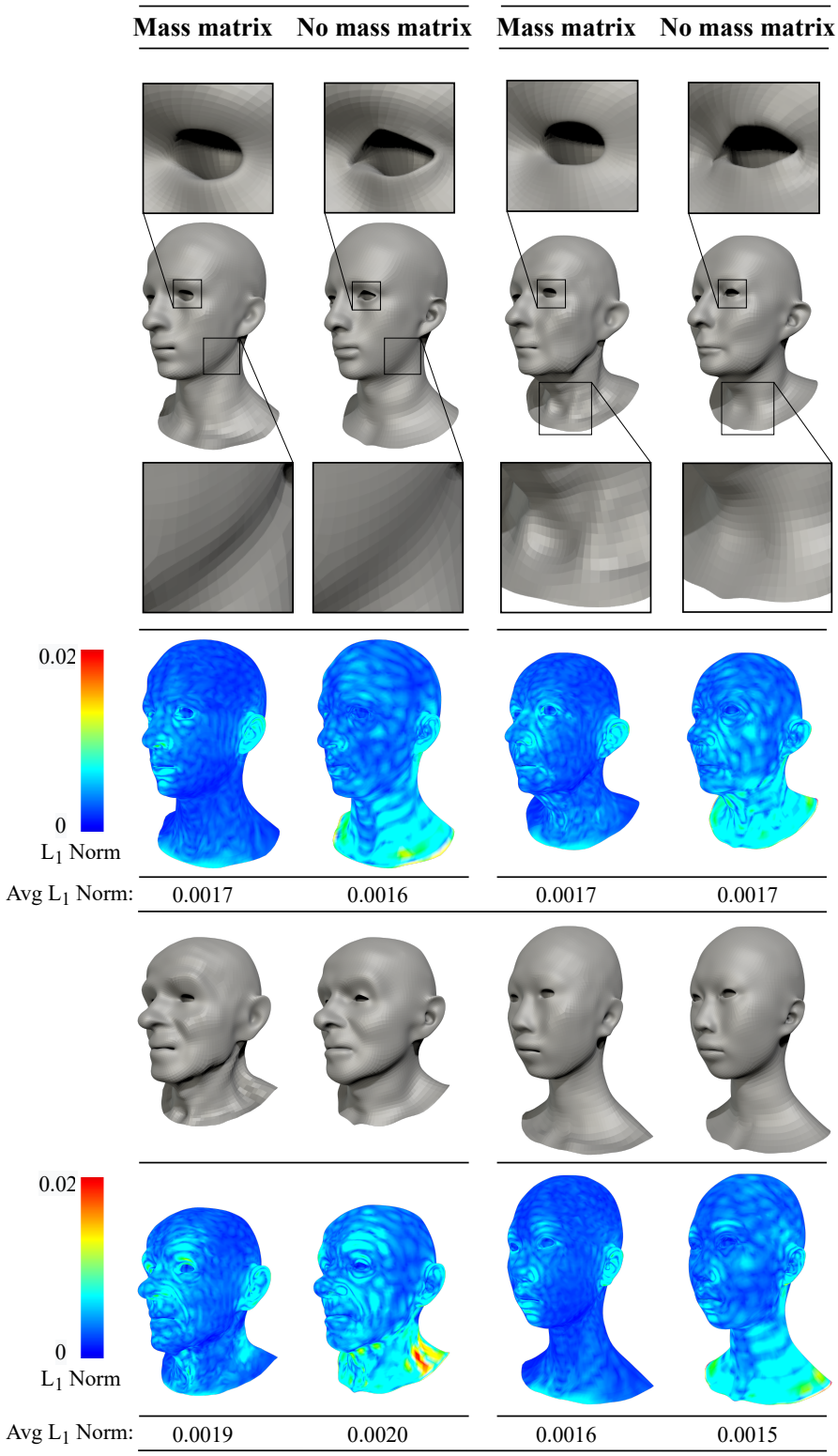


Figure 4.6 Qualitative comparison of spectral mesh decomposition using Equation (4.4) under "Mass matrix" columns and Equation (4.3)) under "No mass matrix" columns. Despite similar ($\pm 10^{-4}$) average L_1 norm between the partitioned mesh and the original signal, the distribution of L_1 norm varies across the vertices. The areas of an eye, neck and jawline are magnified to demonstrate the spatial frequency imbalance in meshes under "No mass matrix" columns comparing to meshes under "Mass matrix" columns.

vertices in $\mathbf{P}_{(500)_{(4.3)}}$. The areas with coarse tessellation, such as the neck, the cheeks and the scalp, lack the lower frequency information which is present in $\mathbf{P}_{(500)_{(4.3)}}$. On the other hand, the areas of the eyes, nose, lips and ears have finer tessellation and $\mathbf{P}_{(500)_{(4.3)}}$ contains higher frequency information in these areas.

The spatial frequency imbalance is demonstrated in $\mathbf{P}_{(500)_{(4.3)}}$ by magnifying the areas of an eye, neck and jawline. In the area of an eye, $\mathbf{P}_{(500)_{(4.3)}}$ contains high frequency signal, such as shape of canthi. Despite more detailed eye, $\mathbf{P}_{(500)_{(4.3)}}$ does not contain lower frequency shape of superficial muscles of the neck, sternal end of the clavicle, jawline and jowls. In contrast, this information is preserved in $\mathbf{P}_{(500)_{(4.4)}}$.

4.3.3 Conclusions

Based on the quantitative and qualitative evaluations, it can be concluded that spectral decomposition of meshes following Equations (4.1) and (4.3) produces partitions with spatial frequency imbalance, as it depends on the density of mesh tessellation. It can be also concluded that using the mass matrix normalisation following Equations (4.2) and (4.4) counters this imbalance. Therefore, the mass matrix hypotheses (1) and (3) are true, and by implication hypotheses (2) and (4) are true. Therefore, Equations (4.10) and (4.11) are chosen to be used in the proposed method.

It is observed that using the mass matrix normalisation requires larger size k of the bandwidth to obtain similar discrepancy between the lower frequency partition and the original signal. Nevertheless, it has been demonstrated that the requirement to use higher value of k does not affect the CPU time, and using the mass matrix normalisation insignificantly increases computational time of preprocessing.

4.4 Deep Spectral Meshes

This section explains the details of our parametric deep face model, which uses the spectral decomposition of meshes. An overview of the model is provided in Figure 4.1.

4.4.1 Vertex Representation

Given a dataset of triangle meshes with shared connectivity, i.e., with the same triangle-vertex index incidence table to associate each triangle with its three bounding vertices, the triangle meshes are translated to make their centre coincide with the origin. Then, their mean is computed, which is an averaged shape of all triangle meshes denoted as $\bar{\mathbf{P}}$. Next, each triangle mesh is translated and rotated to align with the mean. Following this, the

triangle meshes are uniformly scaled to fit within a cube with sides measuring two units each.

Spectral Decomposition

Laplacian matrix $\mathbf{L}$ in Equation (4.2) consists of elements $\mathbf{L}_{ij}$, which are determined by the following cotangent Laplacian operator used in [39]:

$$\mathbf{L}_{ij} = \begin{cases} j \in \mathcal{N}_i & \cot \alpha_{ij} + \cot \beta_{ij} \\ j \notin \mathcal{N}_i & 0 \\ i = j & -\sum_{k \neq j} \mathbf{L}_{ik} \end{cases}, \tag{4.10}$$

where $\mathcal{N}_i$ are vertices ..., $j-1$, j, $j+1$, ... in the one-ring neighbourhood of vertex i and α_{ij} and β_{ij} are the angles opposite to edge ij, as shown in Figure 4.7.

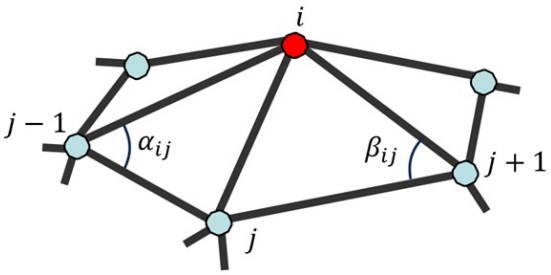


Figure 4.7 One-ring neighbourhood vertices and the angles used to calculate cotangent weights.

For a large triangle mesh, Laplacian matrix $\mathbf{L}$ in Equation (4.2) has many eigenvectors. Calculating all the eigenvectors of such Laplacian matrix is slow. To tackle this problem, only the first k eigenvectors are calculated. Meshes are partitioned into two frequency bands: low and high frequencies. The former band is defined by the first k eigenvectors $\mathbf{U}_{(k)}$, which correspond to the k smallest eigenvalues of Laplacian matrix $\mathbf{L}$. Section 4.4.1 covers the high-frequency band.

We let $\hat{\mathbf{P}}$ be a displacement from a mean so that $\hat{\mathbf{P}} = \mathbf{P} - \bar{\mathbf{P}}$. As eigenvectors $\mathbf{U}_{(k)}$ correspond to the lowest eigenvalues, they form a discrete space of slowly changing values. Therefore, by projecting $\hat{\mathbf{P}}$ onto space $\mathbf{U}_{(k)}$, the resulting vertex positions $\mathbf{P}_{\text{low}}$ are $\hat{\mathbf{P}}$, which is a mean deformed by only low-frequency displacements. $\mathbf{P}_{\text{high}}$ represents the remaining high-frequency displacements of a mean $\bar{\mathbf{P}}$. Meshes $\mathbf{P}_{\text{low}}$ and $\mathbf{P}_{\text{high}}$ are computed as

follows:

$$\begin{aligned} \mathbf{X} &= \mathbf{U}_{(k)}\mathbf{U}_{(k)}^{T}\mathbf{M}, \\ \mathbf{P}_{\text{low}} &= \mathbf{X}\hat{\mathbf{P}} + \bar{\mathbf{P}}, \\ \mathbf{P}_{\text{high}} &= (\mathbf{I} - \mathbf{X})\hat{\mathbf{P}} + \bar{\mathbf{P}}. \end{aligned} \tag{4.11}$$

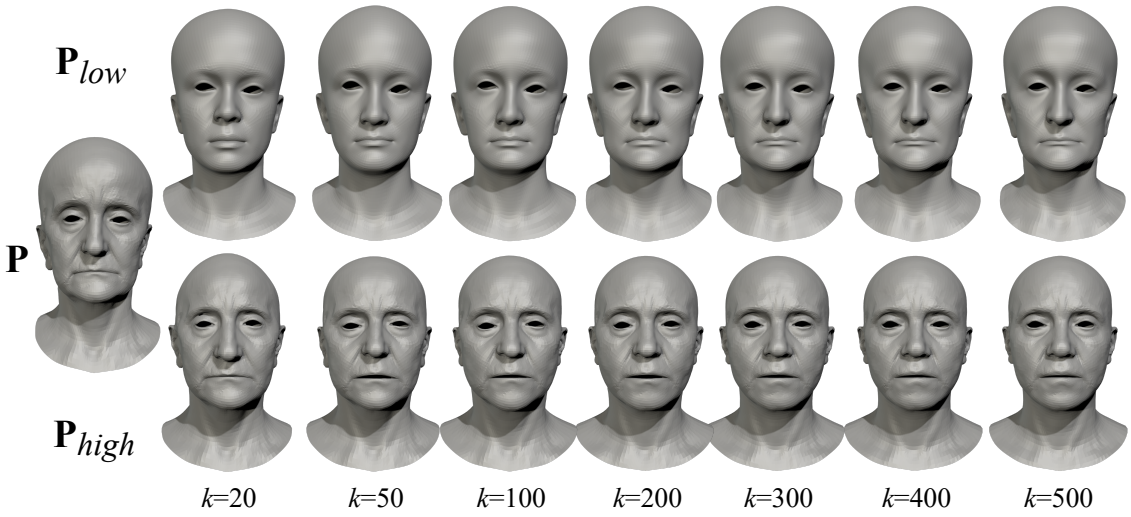


Figure 4.8 Visualisation of meshes $\mathbf{P}_{low}$ and $\mathbf{P}_{high}$ produced using Equation (4.11) from a mesh $\mathbf{P}$ using different parameters k.

High-Frequency Band

High-frequency band $\mathbf{P}_{high}$ is encoded in a normalised deformation representation (DR) [42, 136]. This representation encodes per-vertex deformation gradient $\mathbf{T}_i$ between the position of vertex $\mathbf{p}_i$ on mean $\bar{\mathbf{P}}$ and the position of deformed vertex $\mathbf{p}'_i$ on $\mathbf{P}_{high}$. Following [8, 116], deformation gradient $\mathbf{T}_i$ is calculated by solving the following weighted least-squares system, which minimises energy $E(\mathbf{T}_i)$ such that

$$E(\mathbf{T}_i) = \sum_{j \in \mathcal{N}_i} c_{ij} \left\| (\mathbf{p}'_i - \mathbf{p}'_j) - \mathbf{T}_i(\mathbf{p}_i - \mathbf{p}_j) \right\|^2 , \tag{4.12}$$

where c_{ij} are cotangent weights calculated on a mean of the training meshes.

As linear interpolation of matrices $\mathbf{T}_i$ is meaningless, they are decomposed into a rotational part and a scale/shear part using polar decomposition so that $\mathbf{T}_i = \mathbf{R}_i\mathbf{S}_i$. The rotation matrix $\mathbf{R}_i$, mapped to $\log \mathbf{R}_i$, can be linearly interpolated and then converted back to $\mathbf{R}_i = \exp(\log \mathbf{R}_i)$. Finally, identity matrix $\mathbf{I}$ is subtracted from $\mathbf{S}_i$.

Although both per-vertex matrices $\mathbf{R}_i$ and $\mathbf{S}_i$ contain nine elements each, only six non-trivial scale elements and three non-trivial rotation elements are combined to construct

displacement vectors $\mathbf{f}_i$, where $|\mathbf{f}_i| = 9$. The results of our experiments, shown in Table 4.4, demonstrate that the normalisation of DR results in a lower reconstruction error of test data. Consequently, each channel is normalised to range $[-1, 1]$.

Low-Frequency Band

Meshes $\mathbf{P}_{\text{low}}$ remain in the Euclidean coordinate representation. Based on our experiments in Section 4.7, inputs $\mathbf{F}_{low}$ are standardised by subtraction of the mean and division by standard deviation.

4.4.2 Graph Network Architecture

Our network consists of two fully convolutional variational graph autoencoders, as depicted in Figure 4.1. Encoders E_{high} and E_{low} take as input high-frequency features $\mathbf{F}_{high}$ and low-frequency features $\mathbf{F}_{low}$, respectively (see Section 4.4.1). The encoders compress each input to compact distributions with means μ_{high} and $\mu_{low} \in \mathbb{R}^{8\times4}$, and deviation vectors σ_{high} and $\sigma_{low} \in \mathbb{R}^{8\times4}$. Subsequently, output means and deviations are concatenated so that $\mu = [\mu_{high} \mid \mu_{low}]$ and $\sigma = [\sigma_{high} \mid \sigma_{low}]$.

Latent code $\mathbf{Z} \in \mathbb{R}^{8\times8}$ is obtained in a stochastic process. First, standard deviation σ is multiplied with scalar ε drawn from a normal distribution. Then, the result is added to mean vector μ so that $\sigma \times \varepsilon + \mu = \mathbf{Z} = [\mathbf{Z}_{high} \mid \mathbf{Z}_{low}]$.

Both decoders, E_{high} and E_{low}, take $\mathbf{Z}$ as input. This way, E_{high} decodes high-frequency displacement $\mathbf{F}'_{high}$ from code $\mathbf{Z}_{high}$ conditioned on code $\mathbf{Z}_{low}$. Analogically, E_{low} decodes low-frequency displacement $\mathbf{F}'_{low}$ from code $\mathbf{Z}_{low}$ conditioned on code $\mathbf{Z}_{high}$. Controlling the influence of this conditioning is discussed in Section 4.5.

4.4.3 Network Structure

Both variational graph autoencoders have the same structure. They use convolution/transposed convolution operations and upsampling/downsampling residual layers introduced in [151]. However, other types of graph convolutional and pooling operators can be used instead if they accept input defined solely on vertices. Therefore, the encoders and decoders operating on signal defined on edges would not be suitable, as our method uses low- and high-frequency signal defined on vertices.

The encoders put their inputs through five graph convolutional layers. The local convolution kernels are sampled from the global kernel weight basis to perform convolutional operations on irregular graphs. Therefore, the network learns the shared global kernel weight basis and per-vertex sampling functions [151]. We use stride = 2, kernel radius = 2, weight basis = 35, and channel dimensions = [$|\mathbf{f}|$, 32, 62, 128, 128, 4]. All layers, except the

last one, are followed by the *ELU* activation function. The outputs from the *ELU* are added to outputs from a downsampling residual layer. The decoders mirror the encoders with five blocks of graph transposed convolutional layers followed by the *ELU*s and upsampling residual layers.

4.4.4 Training Process

The two variational graph autoencoders are trained simultaneously. At each iteration, the weights and biases of each encoder–decoder pair are updated through backpropagation using two separate Adam optimisers. Learnable parameters of E_{high} and D_{high} are optimised in terms of loss L_{high}, while parameters of E_{low} and D_{low} are updated in terms of loss L_{low}. Losses are calculated as follows:

$$\begin{aligned} L_{high} = ||\mathsf{denorm}(\mathbf{F}_{high}) - \mathsf{denorm}(D_{high}(\mathbf{Z}))||_1 \\ +\phi\mathsf{KL}(\mathscr{N}(0,1)||p(\mathbf{Z}_{high}|\mathbf{F}_{high})), \end{aligned} \tag{4.13}$$

$$\begin{aligned} L_{low} = ||\mathsf{destd}(\mathbf{F}_{low}) - D_{low}(\mathbf{Z})||_1 \\ +\phi\mathsf{KL}(\mathscr{N}(0,1)||p(\mathbf{Z}_{low}|\mathbf{F}_{low})). \end{aligned} \tag{4.14}$$

In other words, both losses are a sum of two terms. The first one is the L_1 norm of a difference between the ground truth and its reconstruction. The second one is weighted Kullback–Leibler (KL) divergence, which measures a difference between normal distribution $\mathscr{N}(0,1)$ and latent vector distribution, where ϕ is a scalar weight.

Since D_{high} outputs the normalised DR vectors (see Section 4.4.1), the output, $\mathbf{F}'_{high}$, as well as the ground truth, $\mathbf{F}_{high}$, are denormalised before calculating the L_1 norm of a difference between them. Denormalisation is denoted with the $\mathsf{denorm}(\cdot)$ function. In the case of D_{low}, the decoder outputs a destandardised signal in Euclidean coordinates. Therefore, only ground truth $\mathbf{F}_{low}$ is destandardised before computing the L_1 norm term. Here, destandardisation is denoted with the $\mathsf{destd}(\cdot)$ function.

4.4.5 Inference

This section describes the postprocessing steps required to obtain reconstructed vertex positions $\mathbf{P}'$. To be a valid input to the network, every mesh with vertex positions $\mathbf{P}$ must be represented with $\mathbf{F}_{high}$ and $\mathbf{F}_{low}$, following the process described in Section 4.4.1. At inference, the network generates normalised DR vectors $\mathbf{F}'_{high}$ and Euclidean coordinates $\mathbf{F}'_{low} = \mathbf{P}'_{low}$. To calculate $\mathbf{P}'_{high}$, $\mathbf{F}'_{high}$ is denormalised and converted back from the normalised DR representation to Euclidean coordinate representation, as in [136].

Subsequently, $\mathbf{P}'_{high}$ and $\mathbf{P}'_{low}$ are combined in the following way:

$$\mathbf{P}' = (\mathbf{I} - \mathbf{X})\mathbf{P}'_{high} + \mathbf{X}\mathbf{P}'_{low}, \tag{4.15}$$

where $\mathbf{X}$ is the matrix from Equation (4.11).

4.5 Conditioning Influence

Conditioning, described in Section 4.4.2, allows for the network to generate plausible facial meshes from a joint distribution of high- and low-frequency information. However, it can be desirable to tame the influence of conditioning to increase artistic control and prevent overfitting.

In an extreme case, the impact of conditioning could be removed if all the conditioning parameters were set to a constant value, for example, zero. On the other hand, if only a pseudo-randomly drawn fraction of these parameters was set to zero at each training iteration, the impact of conditioning would lower. It is proposed to achieve this effect by applying a dropout [54] to the conditioning parameters.

Specifically, at each iteration, conditioning parameters $\mathbf{Z}_{low}$ of decoder D_{high} are randomly zeroed with probability $(1-\gamma)$ using the samples from the Bernoulli distribution. The same applies to conditioning parameters $\mathbf{Z}_{high}$ of decoder D_{low}. Therefore, scalar γ can be called a Conditioning Factor, where $\gamma = 0$ prevents conditioning and $\gamma = 1$ means full conditioning.

4.6 Implementation Details

Our parametric models are trained with the following datasets of facial meshes: Facsimile ™ [59] (202 meshes, 14,921 vertices each), FaceScape [144] (807 meshes, 26,317 vertices each) and FaceWarehouse [18] (150 meshes, 11,510 vertices each). As all three datasets contain subjects performing a set of facial expressions, only the shapes with a neutral expression are used. All meshes within each dataset have triangular faces with consistent connectivity. In our experiments, each dataset is split into training, validation and test subsets in proportions 85:5:10.

ARPACK [73] is used to solve the generalised eigenvalue problem in Equation (4.2). Only the first k eigenvectors are computed. The networks are trained with $k = 500$ and $k = 1000$, the Conditioning Factor $\gamma = 1.0$ (full conditioning) and $\gamma = 0.4$ (partial conditioning).

The networks are implemented in Pytorch 1.8 [93]. In all experiments, the networks are trained for 500 epochs, with a learning rate of 10^{-4} and a learning rate decay of 1%

after each epoch. KL divergence weight ϕ from Equations (4.13) and (4.14) is 10^{-6}. Adam optimiser's hyper-parameters are set to $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The networks trained with the Facsimile™ and FaceWarehouse datasets are optimised with a batch size of 16, while those trained with FaceScape have a batch size of 8 due to GPU memory constraints.

Matrix $\mathbf{X}$ from Equations (4.11) and (5.10) needs to be pre-computed only once. Otherwise, its frequent computation can significantly lower the performance of data preprocessing. Table 4.1 compares the CPU time and memory required to compute $\mathbf{U}$ and $\mathbf{X}$ on different datasets. Additionally, the table shows the CPU time of the final assembly Equation (5.10). This operation is performed each time a new mesh is inferred and takes 0.20 to 0.98 s per mesh, depending on the dataset used. For faster eigendecomposition of Laplacian $\mathbf{L}$ from Equation (4.10), approximation techniques can be used. Nevertheless, they are not used in our implementation.

Table 4.1 Per-mesh CPU time and CPU memory required to compute terms from Equations (4.11) and (5.10). Three datasets of different vertex count are compared: FaceWarehouse [18] (150 meshes, 11,510 verts), Facsimile ™ [59] (202 meshes, 14,921 verts) and FaceScape [144] (26,317 verts).

	FaceWarehouse (11,510 verts)	**Facsimile™ (14,921 verts)**	**FaceScape (26,317 verts)**
Computation of $\mathbf{U}$ CPU time [s]	26.92	33.72	58.22
Computation of $\mathbf{X}$ CPU time [s]	2.11	7.70	9.88
Computation of Equation (5.10) CPU time [s]	0.20	0.36	0.98
Computation of $\mathbf{U}$ CPU memory [MB]	45.0	58.3	102.8
Computation of $\mathbf{X}$ CPU memory [GB]	1.04	1.74	5.41

All the experiments are trained and inferred on a single NVIDIA GeForce GTX 1080 Ti with 16 GB memory. Data processing is performed on an Intel(R) Xeon(R) CPU E5-1630 v4 running at 8×3.70 GHz using 32 GB RAM. Training times range between 2 h for the FaceWarehouse dataset and 19 h for the FaceScape dataset.

4.7 Applications and Comparisons

The approach proposed in this paper has a large number of applications. Besides those identified in Section 4.8, this section investigates the applications of the proposed approach in mesh reconstruction, mesh interpolation and multi-frequency mesh editing. It also compares the proposed approach with previously published methods.

4.7.1 Mesh Reconstruction

Mesh reconstruction can be used in representation learning, mesh compression, mesh smoothing and fairing. In this subsection, our proposed method is applied to reconstruct 3D meshes for mesh compression. For illustration, all the reconstruction experiments on the Facsimile™ dataset use a compression rate of 0.14%. These reconstruction experiments compare our method and common representations used in other methods: Euclidean coordinates, standardised Euclidean coordinates and the normalised deformation representation (DR).

Implementation of our method follows the description from Sections 4.4 and 4.6, with the Conditioning Factor $\gamma = 1.0$ and frequencies split at $k = 500$. All other representations do not decompose the mesh into displacements with multiple frequencies. Therefore, they are evaluated on the fully convolutional variational graph autoencoder with a single encoder and decoder. The encoder is the same as E_{high} or E_{low}, and the decoder is the same as D_{high} or D_{low}, without the dropout layer. All the experiments encode to latent space $\mathbf{Z}$ of 64 parameters.

Quantitative Evaluation

Two metrics, the point-wise L_1 norm and the Dihedral Angle Mesh Error (DAME) [127], are used to quantitatively assess the quality of meshes generated by our method. The commonly used L_1 norm $||\mathbf{P} - \mathbf{P}'||_1$ between reconstructed Euclidean coordinates $\mathbf{P}'$ and the ground truth $\mathbf{P}$ is chosen due to its sensitivity to overall shape changes. In other words, the L_1 norm is capable of capturing low-frequency error. Nonetheless, it poorly correlates with high-frequency discrepancies perceived by the human visual system [25]. Therefore, the results are also evaluated on a perceptual metric, as described below.

Given that the final meshes generated by our method are viewed by the human observer, the perceptual quality of the results is essential. Quantitative evaluation of perceptual mesh quality is often overlooked in previous work on parametric models with graph networks. In this work, DAME [127] is used to capture high-frequency discrepancies between the reconstructed and the ground truth meshes, and measure the perceptual quality of the generated results. DAME is selected due to its highest correlation with human judgement, measured on the compression task, compared to other common perceptual metrics. Moreover, DAME is dedicated to datasets with shared connectivity [25].

DAME consists of three elements: the difference between oriented dihedral angles, the masking effect and the visibility weighting. The last one is application-specific, as it changes with the viewing angles and the rendering resolution. Therefore, following the recommendation in [127], the visibility term is replaced with triangle areas. Border edges

are ignored in the calculation of DAME, as oriented dihedral angles cannot be calculated on these edges.

The reconstruction results generated with our approach are compared with those output by other methods: Mesh Autoencoder [151], SpiralNet++ [46], Neural 3DMM [14] and FeaStNet [129]. Table 4.2 presents the quantitative outcomes of this comparison. Our proposed method consistently outperforms all counterparts in terms of the perceptual DAME metric across both the Facsimile™ and FaceWarehouse training and test datasets. Regarding point-wise accuracy measured with the L_1 norm, our method outperforms other compared methods on the FaceWarehouse training dataset. However, on the Facsimile™ training and test sets and the FaceWarehouse test dataset, our method performs less favourably than SpiralNet++ [46] and Neural 3DMM [14]. Nonetheless, a comprehensive assessment encompasses both perceptual and point-wise accuracy perspectives, and the perceptual results generated by [14, 46] have significantly higher perceptual DAME error. These quantitative results are further supported by a qualitative user study in Section 4.7.1.

Table 4.2 Numerical comparison of the reconstruction results of the Facsimile™ and FaceWarehouse datasets using our method ($k = 500$, $\gamma = 1$, $\mathbf{Z} = 64$) and four other methods: Mesh Autoencoder [151], SpiralNet++ [46], Neural 3DMM [14] and FeaStNet [129].

	Training		**Test**	
	L_1 **Norm** $\times\mathbf{10^{-3}}\downarrow$	**DAME** $\times\mathbf{10^{-2}}\downarrow$	L_1 **Norm** $\times\mathbf{10^{-3}}\downarrow$	**DAME** $\times\mathbf{10^{-2}}\downarrow$
Facsimile™				
Ours	1.61	**2.76**	6.42	**3.17**
Mesh Autoencoder.	2.60	6.04	8.32	5.81
SpiralNet++	**1.35**	5.35	6.38	4.87
Neural 3DMM	1.71	3.84	**5.95**	3.81
FeaStNet	2.02	5.30	9.07	5.35
FaceWarehouse				
Ours	**0.91**	**1.10**	6.27	**1.29**
Mesh Autoencoder.	2.54	5.27	5.33	5.50
SpiralNet++	1.21	6.06	4.69	5.63
Neural 3DMM	1.58	4.84	**4.02**	4.46
FeaStNet	1.92	6.81	8.17	6.30

In Table 4.3, the L_1 norm and DAME metrics are used to evaluate reconstructed meshes with our proposed approach and the 3D shape representations from other methods. It is

demonstrated that, across the majority of datasets, our method outperforms other methods in reconstructing examples from the training set. Reconstruction of examples seen by the network during training has applications in 3D mesh compression.

Table 4.3 Quantitative comparison of the reconstruction results with our method ($k = 500$, $\gamma = 1$) and with common representations used in other methods: Euclidean coordinates [23, 50, 151], standardised Euclidean coordinates [14, 22, 43, 46, 102] and the normalised deformation representation (DR) [62, 136]. To ensure a fair comparison between our method and other input representations, they are evaluated on the fully convolutional variational graph autoencoder with a single encoder and a single decoder. The encoder is the same as E_{high} or E_{low}, and the decoder is the same as D_{high} or D_{low}, without the dropout layer. All the comparisons encode to latent space $\mathbf{Z}$ of 64 parameters. Our method outperforms the reconstruction of examples from the training set on most datasets. On the test set, our method favourably compromises between the point-wise L_1 precision and the perceptual DAME metric. Other methods considerably sacrifice one of these in favour of another.

	Training		**Test**	
	L_1 **Norm** $\times\mathbf{10^{-3}}\downarrow$	**DAME** $\times\mathbf{10^{-2}}\downarrow$	L_1 **Norm** $\times\mathbf{10^{-3}}\downarrow$	**DAME** $\times\mathbf{10^{-2}}\downarrow$
Facsimile™				
Ours	**1.61**	**2.76**	6.42	3.17
DR	4.77	3.05	9.29	**3.00**
Euclidean Std.	2.36	4.90	**5.78**	3.89
Euclidean	2.60	6.04	8.32	5.81
FaceWarehouse				
Ours	**0.91**	**1.10**	6.27	1.29
DR	2.20	1.14	7.42	**1.22**
Euclidean Std.	1.11	3.23	**5.33**	2.53
Euclidean	2.54	5.27	5.34	5.50
FaceScape				
Ours	1.27	2.25	1.65	2.41
DR	6.06	**1.64**	5.92	**1.61**
Euclidean Std.	**0.96**	1.81	**1.30**	1.82
Euclidean	1.32	2.20	1.71	2.26

The reconstructions of meshes from the test set reveal a pattern. The standardised Euclidean representation achieves the lowest L_1 norm error, and the normalised

DR has the lowest DAME error. Nonetheless, in the case of the Facsimile™ [59] and FaceWarehouse [18] datasets, Euclidean and standardised Euclidean coordinates perform the worst on the DAME metric. Analogously, DR has the highest L_1 norm. It can be concluded that, with these representations, either point-wise precision or perceptual quality must be sacrificed significantly.

Our method balances these metrics favourably, as demonstrated in Figure 4.9. The results on the Facsimile™ dataset reveal that, compared to normalised DR, our method achieves a 30.9% lower L_1 norm error, with only a 5.7% increase in DAME error. Contrasted with standardised Euclidean coordinates, our approach yields an 18.5% lower DAME error at the cost of an 11.5% higher L_1 norm error. Now, turning to the FaceWarehouse [18] dataset, our method has a 15.5% lower L_1 norm error than normalised DR, with just a 5.7% higher DAME error. In comparison to standardised Euclidean coordinates, our method results in a 49% lower DAME error at the cost of just a 17.6% higher L_1 norm error.

However, for the FaceScape [144] dataset, our method does not improve upon the reconstruction results from other representations on both training and validation sets. Scatter plots in Figure 4.9 reveal the reason. The proposed approach can improve the quality of reconstructed meshes by utilising dedicated mesh representations for each spectral band. Nevertheless, on the FaceScape dataset, the DR representation yields little improvement of the perceptual quality compared to standardised Euclidean representation. Therefore, our proposed method cannot benefit from using different representations in this particular case. Experiments on the FaceScape dataset demonstrate that our method can elevate the overall quality of reconstructed meshes only when there is a substantial difference between the point-wise and the perceptual accuracy yielded by at least two different mesh representations.

Table 4.4 compares the deformation representation (DR) with and without the normalisation preprocessing step described in Section 4.4.1. Moreover, the table contrasts Euclidean coordinates with and without the standardisation preprocessing step, as described in Section 4.4.1. Standardisation of low-frequency inputs represented in Euclidean coordinates improves training and validation results across all datasets in the comparison. Normalising high-frequency inputs represented with DR improves the L_1 norm on validation sets and yields a similar or a lower DAME error. The impact of normalising DR is not conclusive on the training dataset, as normalisation of DR improves the L_1 norm and DAME metrics on the FaceWarehouse dataset, while DR without normalisation yields better results on the Facsimile™ dataset.

Figures 4.11 and 4.12 compare the effect of different values of k on perceptual and spatial reconstruction error.

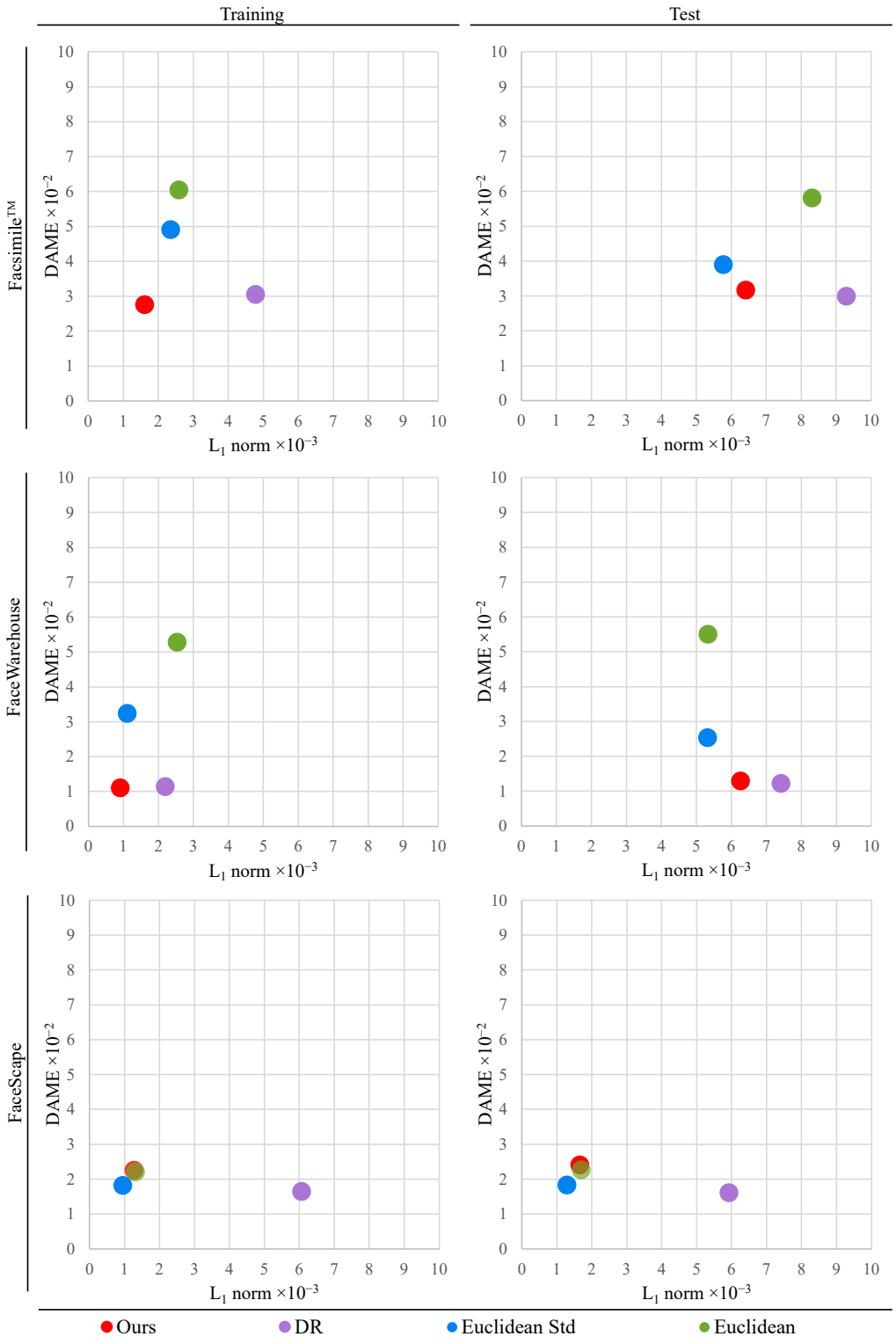


Figure 4.9 Comparison of the reconstruction results with our method ($k = 500$, $\gamma = 1$, $\mathbf{Z} = 64$) and with common representations used in other methods: Euclidean coordinates [23, 50, 151], standardised Euclidean coordinates [14, 22, 43, 46, 102] and normalised deformation representation (DR) [62, 136]. Across Facsimile and FaceWarehouse datasets, our method outperforms in reconstructing examples from the training set and favourably balances perceptual and geometric quality on the Pareto-front of optimal solutions. Our method underperforms on the FaceScape [144] dataset because the benefit of using normalised DR representation for high-frequency information is minuscule compared to standardised Euclidean representation.

Table 4.4 Ablation study on the reconstruction task demonstrating the impact of normalisation of the deformation representation (DR) and the standardisation of Euclidean coordinate inputs.

	Training		**Validation**	
	L_1 **Norm** $\times\mathbf{10^{-3}}\downarrow$	**DAME** $\times\mathbf{10^{-2}}\downarrow$	L_1 **Norm** $\times\mathbf{10^{-3}}\downarrow$	**DAME** $\times\mathbf{10^{-2}}\downarrow$
Facsimile™				
DR without normalisation	**4.01**	**2.77**	12.05	3.07
DR with normalisation	4.77	3.05	**9.29**	**3.00**
Euclidean without standardisation	2.60	6.07	8.32	5.81
Euclidean with standardisation	**2.36**	**4.90**	**5.78**	**3.89**
FaceWarehouse				
DR without normalisation	2.57	1.71	10.03	**1.19**
DR with normalisation	**2.21**	**1.14**	**7.42**	1.22
Euclidean without standardisation	2.54	5.27	**5.33**	5.50
Euclidean with standardisation	**1.12**	**3.23**	**5.33**	**2.53**

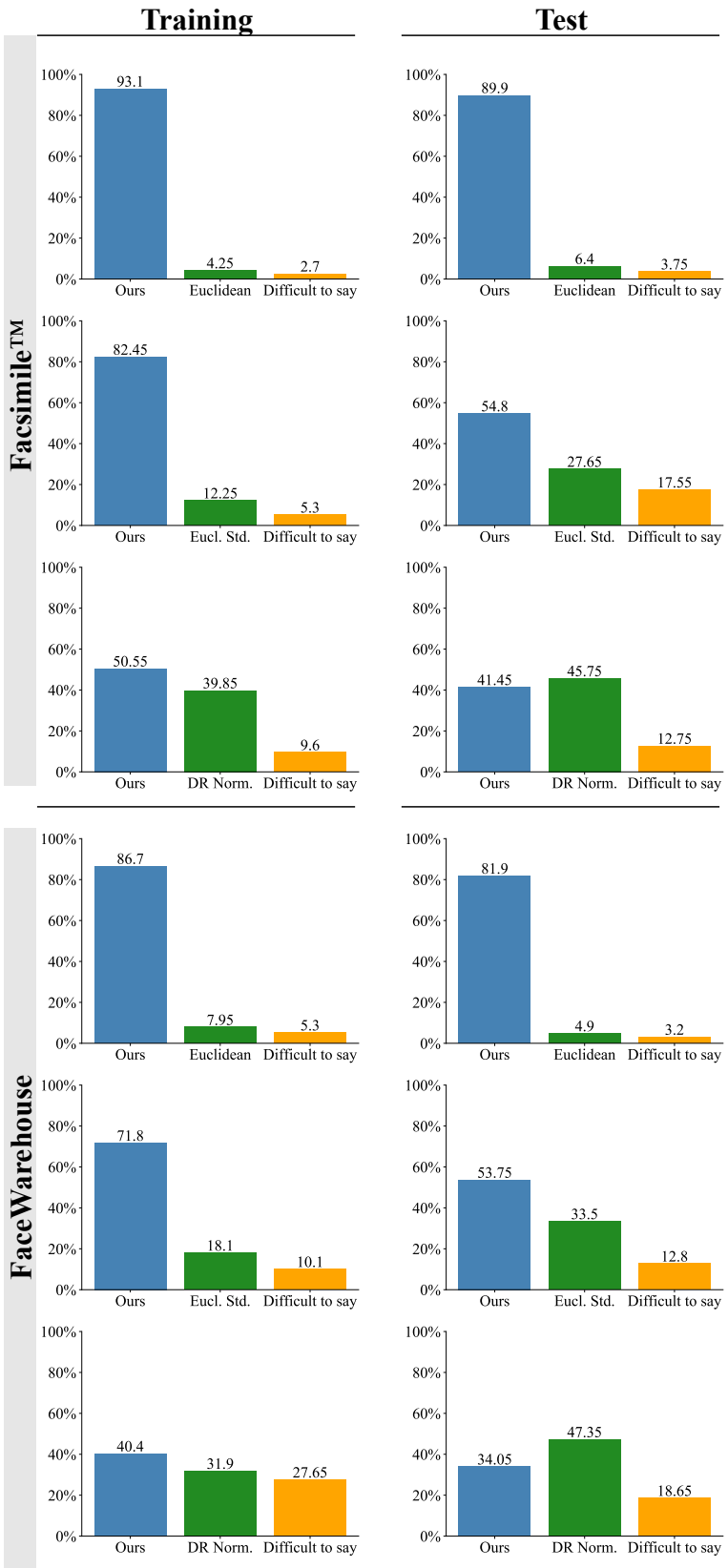


Figure 4.10 The results from the user study comparing our method with common representations used in other methods: Euclidean coordinates [23, 50, 151], standardised Euclidean coordinates (Eucl. Std.) [14, 22, 43, 46, 102] and the normalised deformation representation (DR Norm.) [62, 136]. The bars show the percentage of participants who selected the mesh generated by the given method as more similar to the ground truth mesh. The participants were asked to select "Difficult to say" only when they had to guess between the generated models.

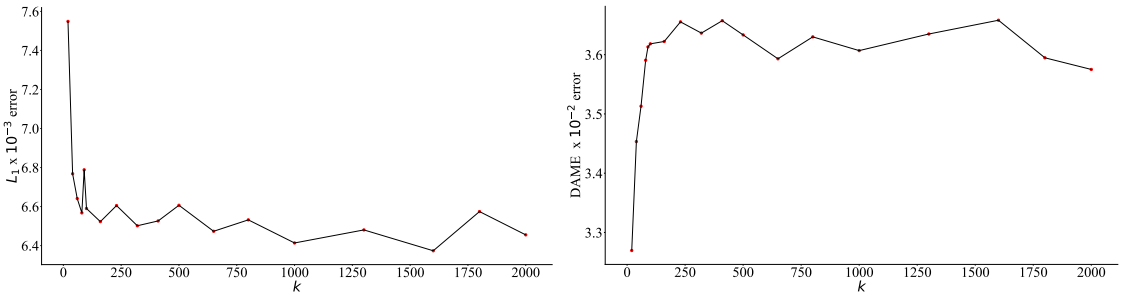


Figure 4.11 Comparison of the impact of 18 different parameters k on (A) L_1 norm and (B) DAME test reconstruction error on the Facsimile™ test dataset.

Qualitative Evaluation

Figures 4.13 and 4.14 provide a visual assessment of the mesh quality in the reconstruction experiments, comparing our method with other commonly used representations. The meshes generated using Euclidean and standardised Euclidean representations display visible surface artefacts and struggle to capture high-frequency details. The severity of surface discrepancies corresponds with the colour visualisation of the DAME error. Results obtained with the normalised deformation representation (DR) are perceptually more similar to the ground truth meshes. Nonetheless, there is noticeable volume loss in the neck, chin and cheeks. In contrast, our method produces results that are perceptually similar to ground truth meshes without experiencing noticeable volume changes.

Figure 4.15 compares the rendered meshes from the Facsimile™ and FaceWarehouse datasets reconstructed with our proposed method and other popular methods: Mesh Autoencoder [151], SpiralNet++ [46], Neural 3DMM [14] and FeaStNet [129]. A user study was conducted to qualitatively assess the perceptual quality of meshes generated by our method ($k = 500$, $\gamma = 1$, $\mathbf{Z} = 64$) compared with meshes synthesised using alternative methods and common representations. The evaluation focused on reconstructions of training and test subsets of the Facsimile™ and FaceWarehouse datasets. The study was conducted online on a representative sample of 94 participants (42 female, 51 male, 1 non-binary), with the following age distribution: 25 participants aged 18–25, 52 aged 26–35, 11 aged 36–45 and 6 aged 46–55. Participants viewed three images of rendered 3D models and were instructed to compare the ground truth reference model (always in the middle) against models A (left) and B (right). Subsequently, participants selected the model they perceived as more similar to the reference model (either option “A” or “B”). In cases where a clear distinction was challenging, participants had the option to choose “Difficult to say”. Each participant completed a total of 48 comparisons. In each comparison, one 3D model was generated by our method, while the other was produced

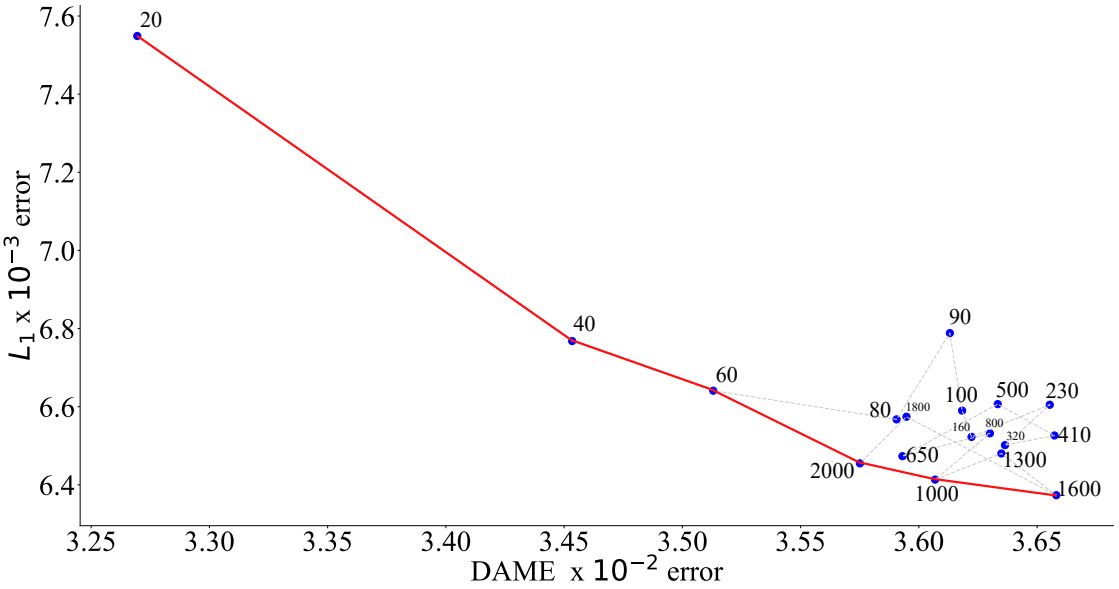


Figure 4.12 Comparison of the effect of different values of k (blue points) on perceptual and spatial reconstruction error. The parameter k affects the trade-off between the perceptual error measured with DAME and spatial fidelity measured with L_1 norm. Values of k which form a Pareto front of optimal solutions when considering solely perceptual quality and spatial fidelity are connected with a red line.

by an alternative method or using a different commonly used representation. Participants were instructed to use a desktop monitor or a tablet and complete the study in a full-screen view.

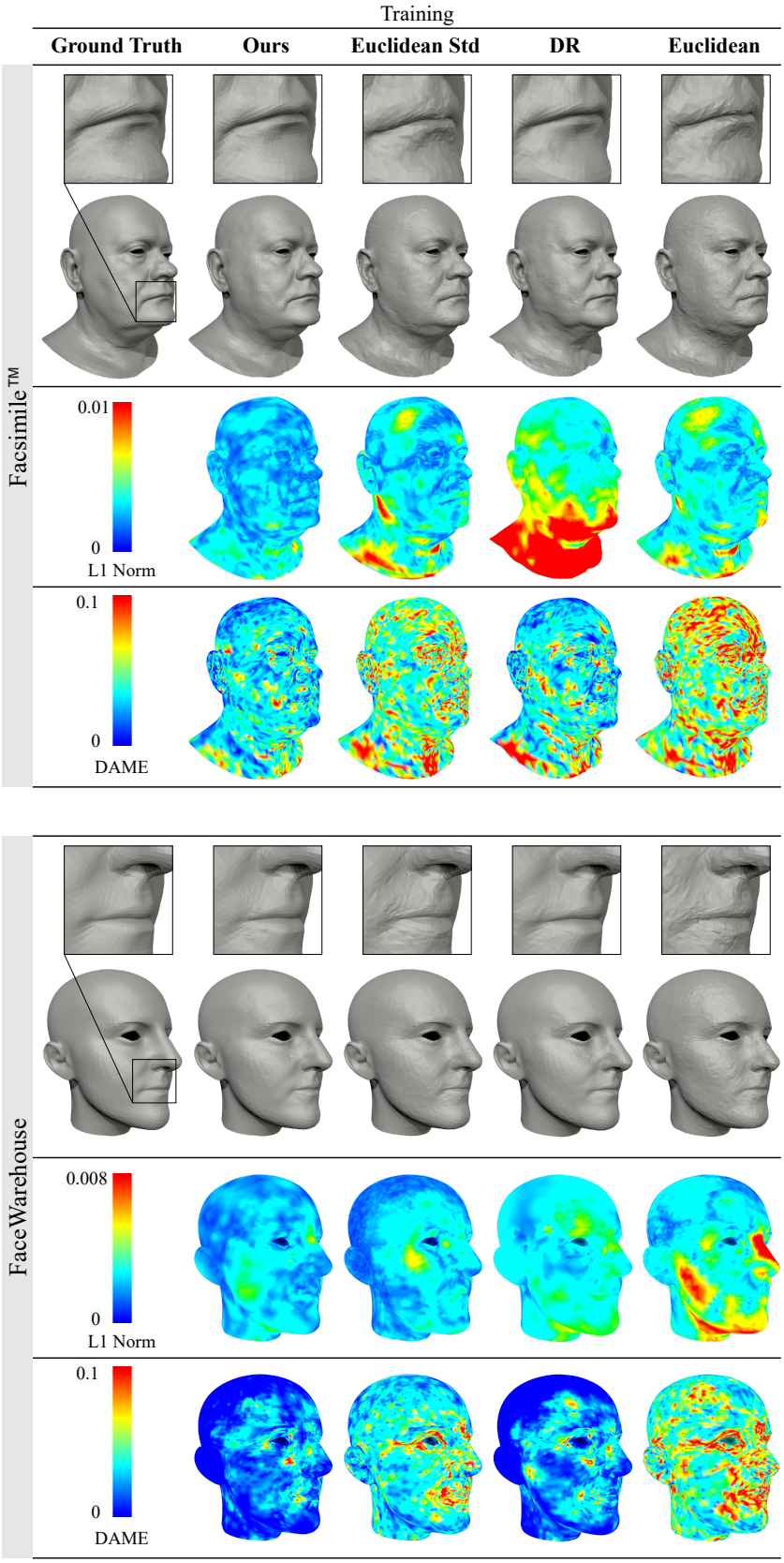


Figure 4.13 Qualitative comparison of the reconstruction results of training data with our method ($k = 500$, $\gamma = 1$) and with common representations used in other methods: Euclidean coordinates [23, 50, 151], standardised Euclidean coordinates [14, 22, 43, 46, 102] and the normalised deformation representation (DR) [62, 136]. The meshes generated by our method achieve superior results compared to other representations. Zooming into the digital version is recommended to see the surface artefacts on the results generated with Euclidean and standardised Euclidean representations.

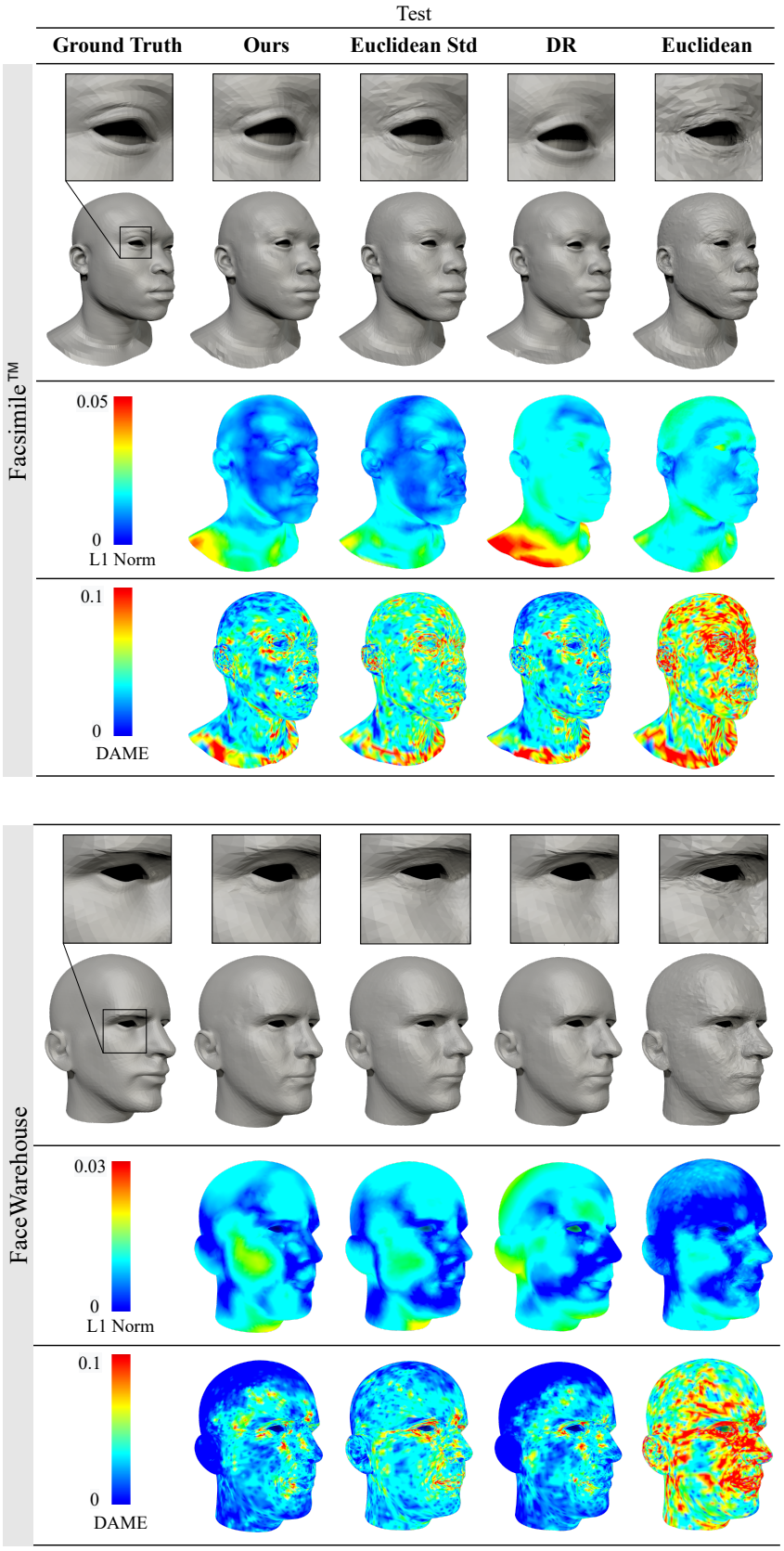


Figure 4.14 Qualitative comparison of the reconstruction results of test data with our method ($k = 500$, $\gamma = 1$) and with common representations used in other methods: Euclidean coordinates [23, 50, 151], standardised Euclidean coordinates [14, 22, 43, 46, 102] and the normalised deformation representation (DR) [62, 136]. The meshes generated by our method have similar surface quality to the outputs with DR while achieving much lower volume loss in the neck, chin, and cheek areas.

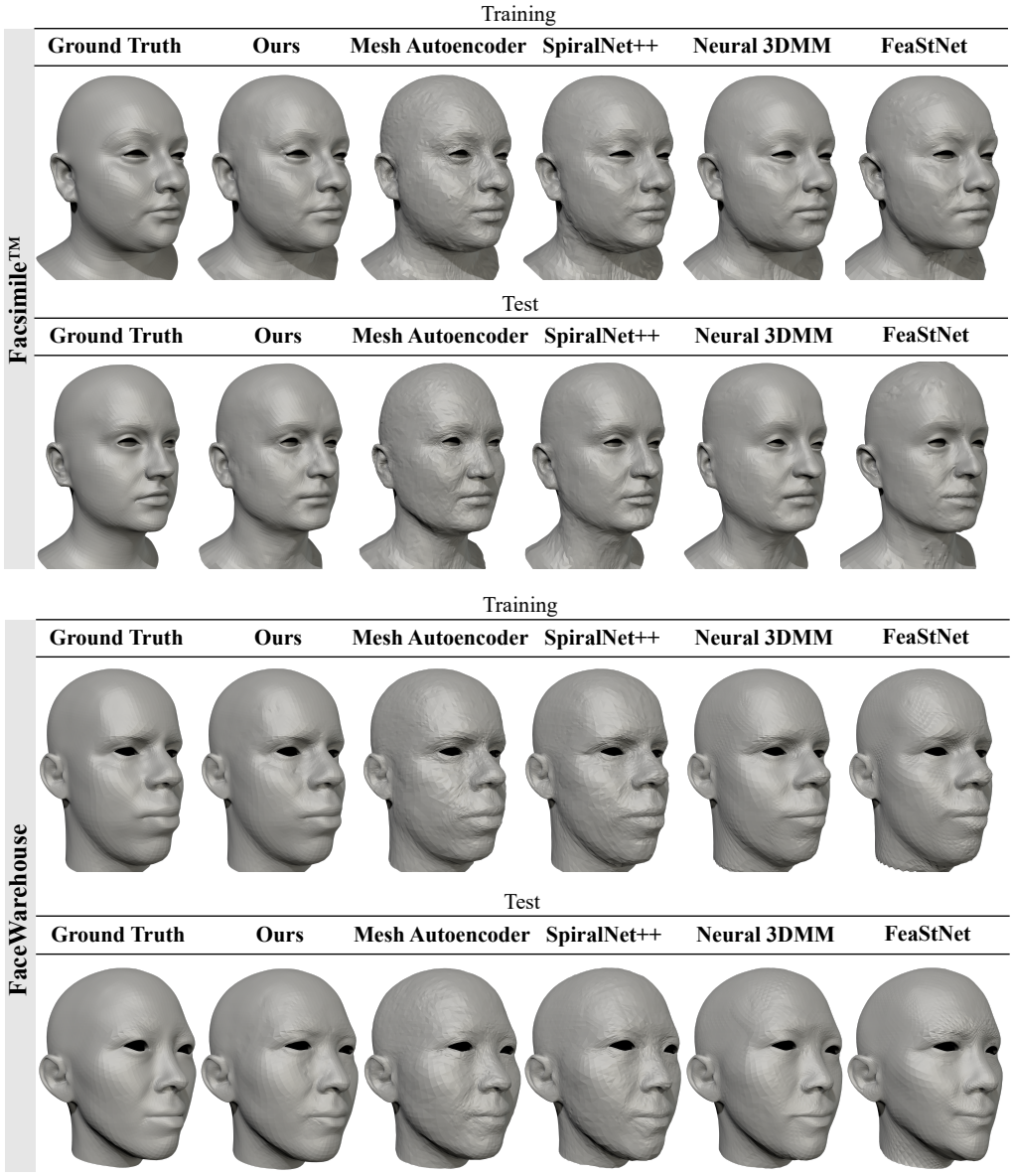


Figure 4.15 Visual comparison of the reconstruction results of the Facsimile™ and FaceWarehouse datasets using our method ($k = 500$, $\gamma = 1$, $\mathbf{Z} = 64$) and four other methods: Mesh Autoencoder [151], SpiralNet++ [46], Neural 3DMM [14] and FeaStNet [129]. It is recommended to zoom into the digital version to compare the reconstructed meshes.

Figure 4.16 displays the aggregated responses from the user study, which compared the visual similarity of the meshes generated by our proposed method with those produced by other methods [151, 46, 14, 129]. In all cases, a strong majority of participants perceived the meshes generated by our method as more similar to the reference than those generated by other compared methods. Participants expressed a high certainty of their responses, with only a median of 4.3% choosing the "Difficult to say" option.

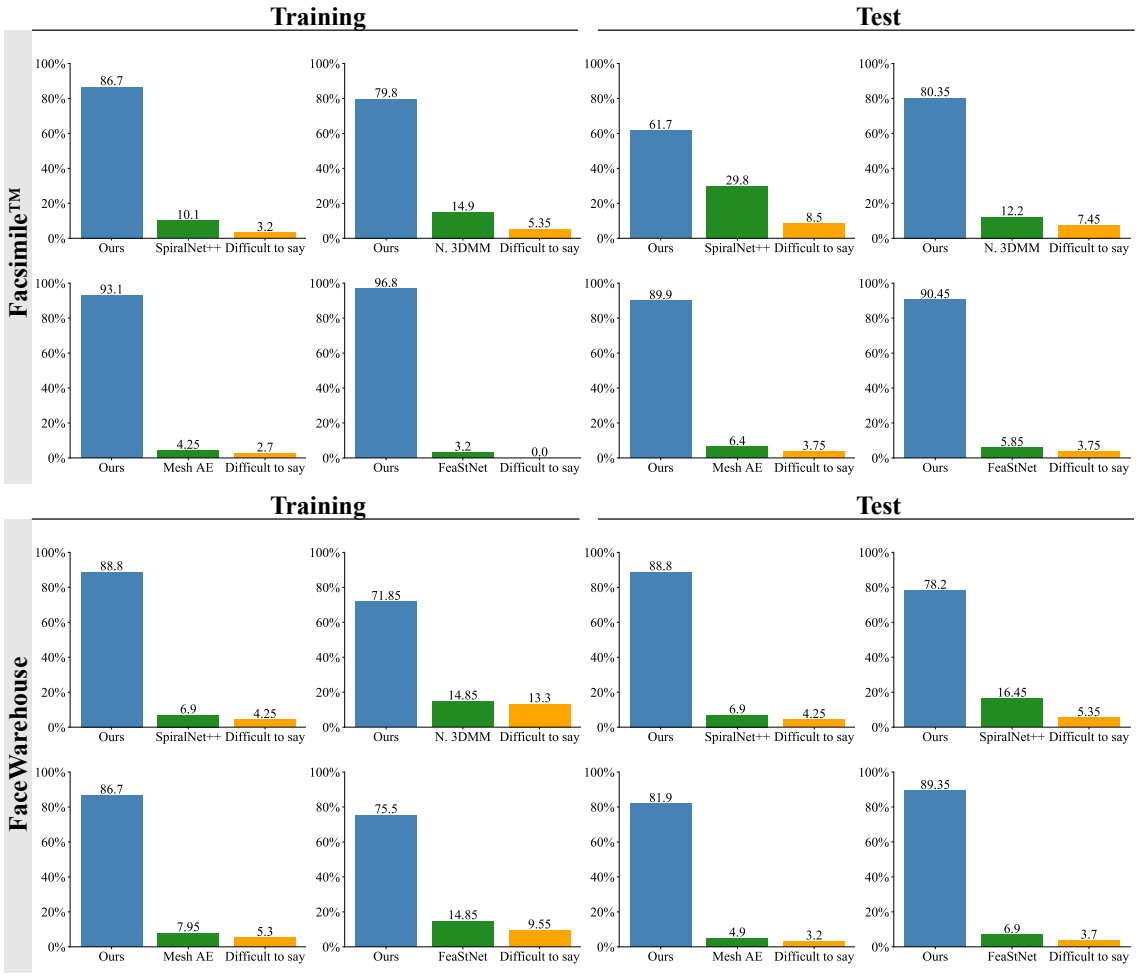


Figure 4.16 The outcomes of the user study, which compared the visual similarity to the ground truth of the meshes generated by our method and other methods: Mesh Autoencoder [151], SpiralNet++ [46], Neural 3DMM [14] and FeaStNet [129]. The bars show the percentage of participants who selected the mesh generated by the given method as more similar to the ground truth mesh. The participants were asked to select "Difficult to say" only when they had to guess between the generated models.

Figure 4.10 presents the results from the user study comparing our method with common representations used in other methods: Euclidean coordinates [23, 50, 151], standardised Euclidean coordinates [14, 22, 43, 46, 102] and the normalised deformation representation (DR) [62, 136]. Participants' responses align with our quantitative analysis of perceptual DAME error from Table 4.3. On average, 87.9% and 65.7% of participants perceived our methods' results as more similar than those from methods using Euclidean coordinates and standardised Euclidean coordinates, respectively. Regarding the comparison with normalised DR representation, the participants were divided. The training sets reconstructed with our method received 10.7 and 8.5 more percentage points of participants' preference. In comparison, the meshes produced with the normalised DR representation

were chosen as more similar on the test sets by 4.3 and 13.3 more percentage points of participants. Consequently, the perceptual similarity of meshes generated by our method and the normalised DR representation is comparable, while our method yields significantly lower point-wise accuracy error, as demonstrated in quantitative analysis.

4.7.2 Mesh Interpolation

Mesh interpolation is widely applied in facial animation to produce new facial meshes from two known facial meshes. In contrast to existing interpolation methods that interpolate two facial meshes, our proposed approach interpolates low- and high-frequency parts of two facial meshes.

In Figure 4.17, the subject in the green outline is encoded into parameters $[\mathbf{Z}_{1l}|\mathbf{Z}_{1h}]$ and the subject in the purple outline is encoded into parameters $[\mathbf{Z}_{2l}|\mathbf{Z}_{2h}]$. High-frequency weight α and low-frequency weight β are used to interpolate between these latent parameters so that the interpolated latent parameters are

$$\mathbf{Z}_{\alpha,\beta} = [(1-\beta)\mathbf{Z}_{1l} + \beta\mathbf{Z}_{2l}|(1-\alpha)\mathbf{Z}_{1h} + \alpha\mathbf{Z}_{2h}]. \tag{4.16}$$

The meshes resulting from Equation (4.16) are shown in Figures 4.17(A), 4.17(B) and 4.18. In the figures, the meshes arranged in a grid are decoded from latent parameters $\mathbf{Z}_{\alpha,\beta}$. When using the linear interpolation in the vertex space between the mesh in the green outline and the mesh in the purple outline, the interpolation equation is

$$\mathbf{P}_\delta = \mathbf{P}_1 + \delta(\mathbf{P}_2 - \mathbf{P}_1), \tag{4.17}$$

where $\mathbf{P}_1$ and $\mathbf{P}_2$ are the vertex coordinates of the meshes in the green outline and the purple outline, respectively, and $0 \leq \delta \leq 1$.

The meshes obtained from Equation (4.17) are shown in Figure 4.17C. Interpolating low- and high-frequency latent parameters with two different conditioning values, 0.4 and 1.0, generates 28 new meshes. In contrast, interpolating the meshes in the green and purple outlines creates only two new meshes.

Our discussion indicates that multi-frequency interpolation noticeably raises the capacity to create novel facial meshes from two known facial meshes. In addition, Figure 4.17 demonstrates the disentanglement of low and high frequencies in the parametric space and illustrates the impact of the Conditioning Factor γ.

When $\gamma = 1.0$, the high-frequency parameters of an elderly subject influence mid-frequencies to impose the generation of plausible faces. However, due to bias towards younger subjects in the dataset, this conditioning significantly restricts the domain of generated faces. For an artist, such editing behaviour may be undesirable. In contrast,

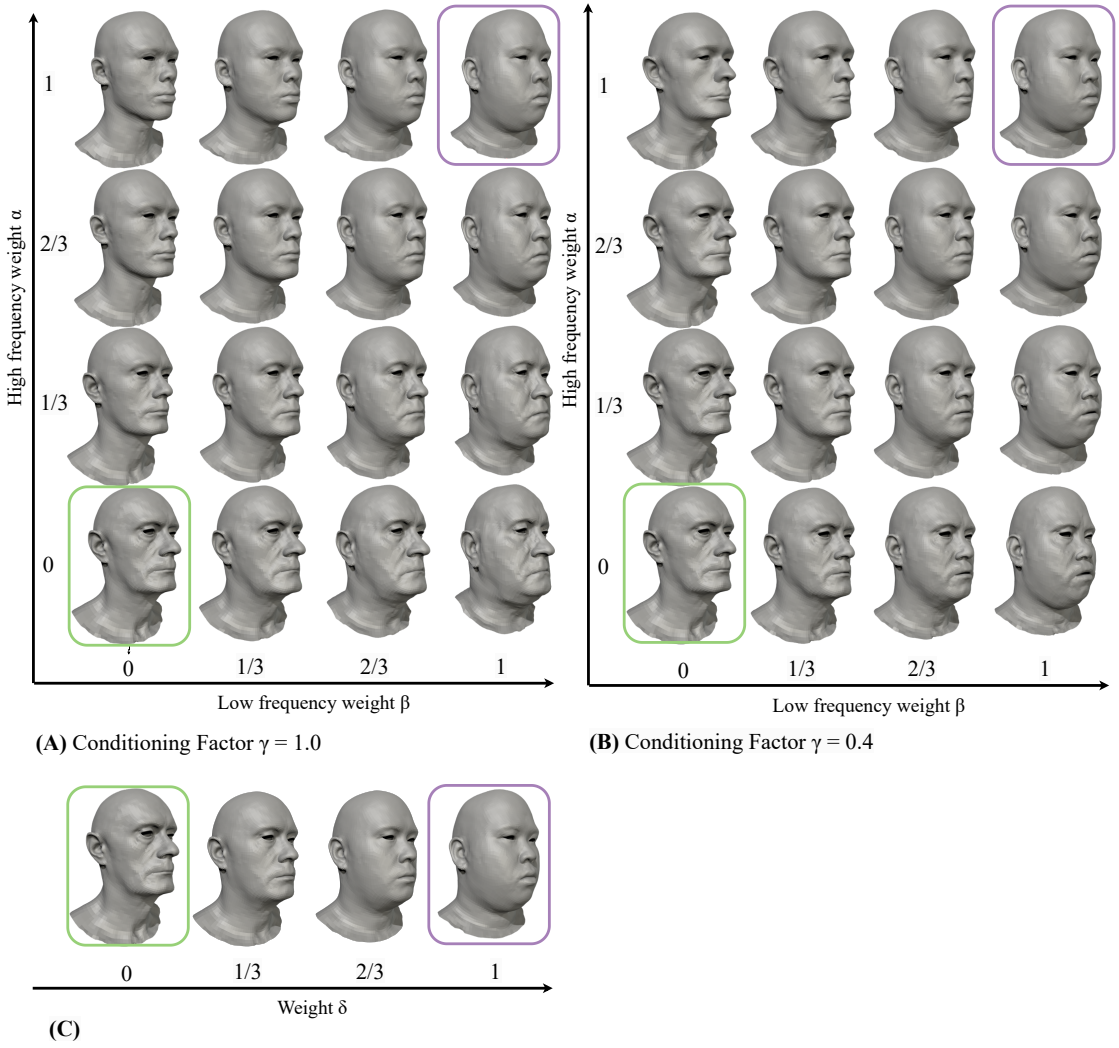


Figure 4.17 Interpolation of low-frequency and high-frequency latent parameters, $k = 500$. Two facial meshes (in green and purple outlines) are encoded. They are from the Facsimile™ [59] dataset. In (A), the model is trained with the Conditioning Factor $\gamma = 1.0$. In (B), the Conditioning Factor $\gamma = 0.4$. The meshes arranged in a grid are decoded from interpolated latent parameters. In (C), the meshes in green and purple outlines are interpolated in the vertex space.

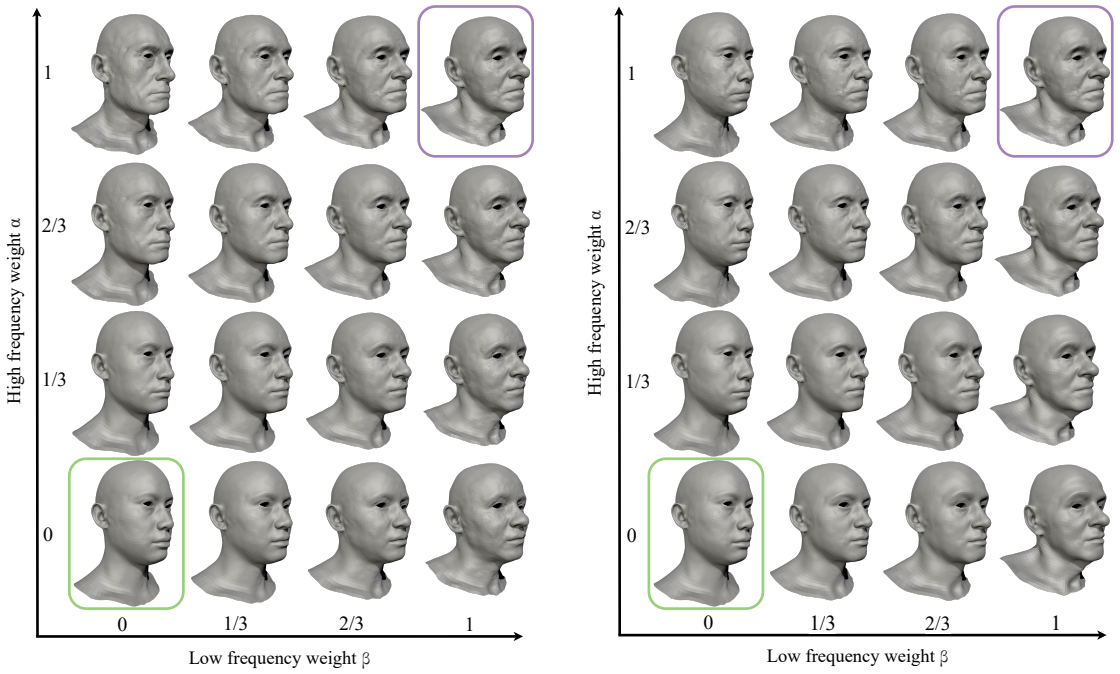
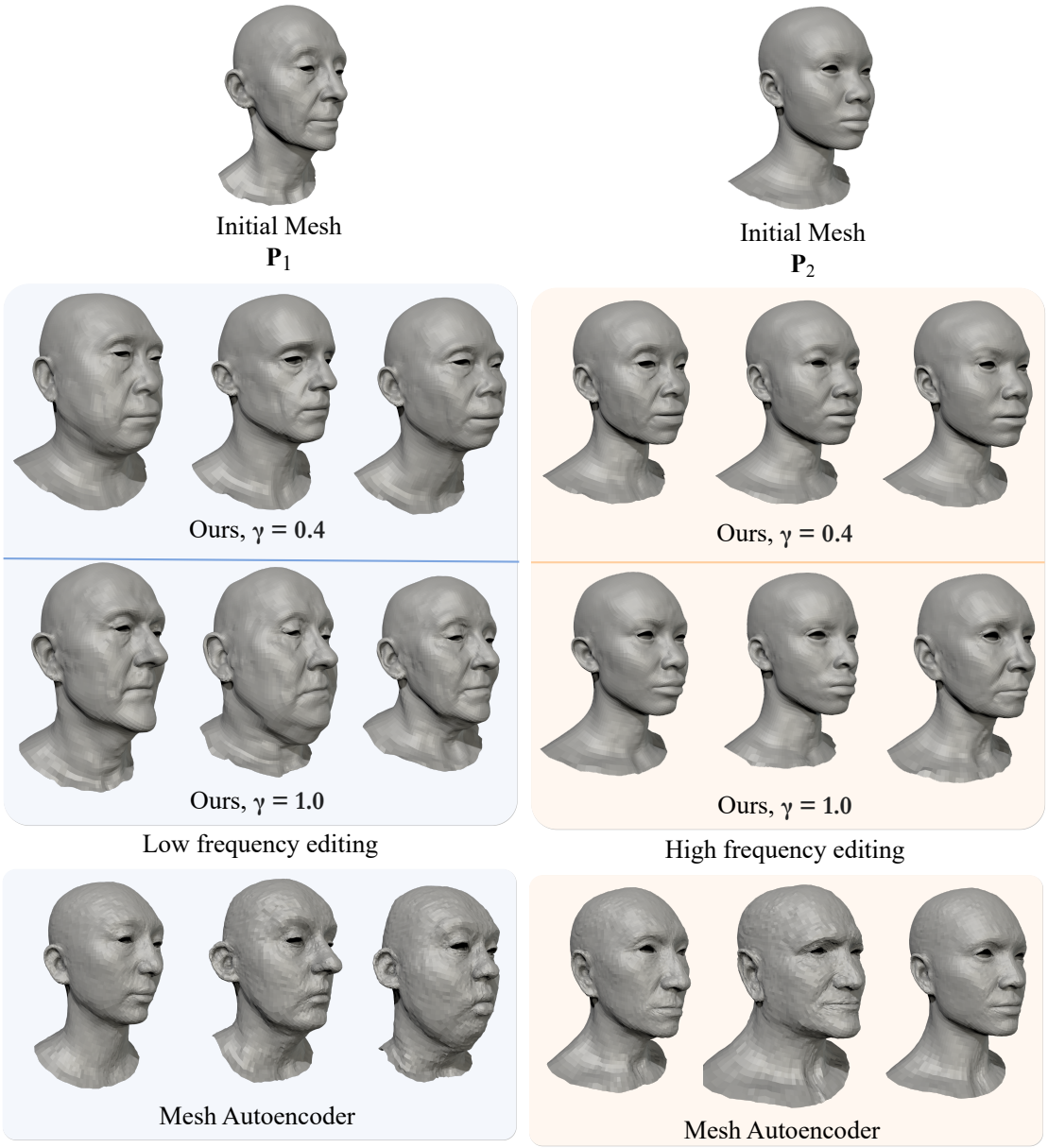


Figure 4.18 Interpolation of low-frequency and high-frequency latent parameters. This time, the parameter $k = 1000$. Two facial meshes (in green and purple outlines) are encoded, both from Facsimile™ [59] dataset. In (A), the model is trained with the Conditioning Factor $\gamma = 1.0$. In (B), the Conditioning Factor $\gamma = 0.4$. The meshes arranged in a grid are decoded from interpolated latent parameters. They demonstrate that with $\gamma = 0.4$, the network can generate implausible examples, such as the older man with young, smooth skin at $(\alpha = 0, \beta = 1)$. This implausible effect is countered with higher Conditioning Factor $\gamma = 1.0$ at $(\alpha = 0, \beta = 1)$.

interpolation of low- and high-frequency parameters with $\gamma = 0.4$ is more predictable, as high- and low-frequency displacements are almost fully disentangled. Nevertheless, plausible faces exist within a joint distribution of low- and high-frequency displacements. Consequently, the network may generate implausible examples, like the older man with young, smooth skin depicted in Figure 4.17B at ($\alpha = 1, \beta = 0$). Therefore, the choice of γ depends on application-specific requirements, whether prioritising more precise and flexible artistic control or the ability to generate only plausible faces.

4.7.3 Multi-Frequency Editing

Figure 4.19 presents the application of our method in editing low- and high-frequency displacements independently. Additionally, the editing capabilities of our model are compared with the method in [151], which does not disentangle low- and high-frequency parameters. In our approach, low-frequency parameters affect the overall head shape while preserving high-frequency details. The Conditioning Factor γ impacts the nature of editing. With $\gamma = 1.0$ (full conditioning), high frequencies condition low-frequency displacements to a more narrow domain of only plausible faces. In contrast, when $\gamma = 0.4$ (partial conditioning), edited heads are more diverse, despite some falling outside of the domain of real faces. When editing high-frequency parameters, the overall head shape remains unchanged, with only fine details being affected. Notably, with $\gamma = 0.4$, the editing of high frequencies becomes more precise and predictable since high- and low-frequency parameters barely influence each other.



(A) Comparison between low-frequency editing of the method and the latent code editing of Mesh Autoencoder.

(B) Comparison between high-frequency editing of the method and the latent code editing of Mesh Autoencoder.

Figure 4.19 Comparison of latent code editing between the proposed method and Mesh Autoencoder [151]. In (**A**), the editing of low-frequency latent codes of encoded mesh $\mathbf{P}_1$. In (**B**), the editing of high-frequency latent codes of encoded mesh $\mathbf{P}_2$. Top and middle row: the examples decoded using our model with $k = 500$ and Conditioning Factor $\gamma = 0.4$ and $\gamma = 1.0$. Bottom row: the results of editing a subset of latent parameters using the method in [151]. The parameters of our method successfully disentangle high and low frequencies. While subjective, it can be observed that lower γ provides more control and produces more diverse results. Meanwhile, altering the parameters of Mesh Autoencoder [151] affects the entire frequency spectrum.

4.8 Conclusions

The methods proposed in this chapter built upon the conclusions from Chapter 3 and introduced Deep Spectral Meshes, an approach that involves the spectral decomposition of 3D meshes in representation learning with graph convolutional networks. It was described how the spatial frequency theory of perception inspired the Deep Spectral Meshes approach. The input representations and the neural network have been described. The implementation details were provided and the applications of the proposed method were investigated. Moreover, this chapter provided a comparison of the proposed approach with previously published methods both quantitatively and qualitatively. The ability to independently edit low and high-frequency displacements of facial meshes was demonstrated using multiple examples. The impact of the Conditioning Factor γ was further explored. γ balances mutually exclusive objectives of independent control of deformations at different frequencies, and generation of plausible synthetic examples. The comparisons between the proposed approach and previously published methods demonstrate the improvements of the proposed method over the state-of-the-art results according to both L_1 and perceptual metric evaluations.

This page is intentionally left blank.

CHAPTER5

PERSONALISED EXPRESSIONS GENERATION

5.1 Introduction

In video games and VFX industry, facial rigs are most commonly based on linear combinations of blendshapes which represent single action units (AUs) and their common combinations. Therefore, building facial rigs requires time and expertise of 3D artists to model often hundreds of blendshapes for a given 3D character identity. As an alternative to this costly and laborious process, the Deformation Transfer (DT) [120] method is used to semi-automatically transfer facial expressions from source identity to target identity. However, shapes produced by DT are not personalised and deformed target identities exhibit features from the source identity, as evidenced in this chapter.

To tackle this problem, the Facsimile dataset of neutral faces coupled with expressions can be used to predict more personalised expressions. It can be hypothesised that:

1. Using a graph variational autoencoder to find a mapping between neutral face identity and its personalised expression can improve the perceptual and geometric accuracy of synthesised expressions comparing to the Deformation Transfer method.

2. The two-step method, which models the mapping between latent space of neutral faces and the personalised expressions, yields lower perceptual and geometric error on test dataset than the end-to-end method, which uses an autoencoder to directly map between neutral face shapes and personalised expressions.

3. Using spectral mesh processing to combine low-frequency deformation synthesised with graph variational autoencoder using global, coordinates-based representation and high-frequency deformation synthesised with graph variational autoencoder

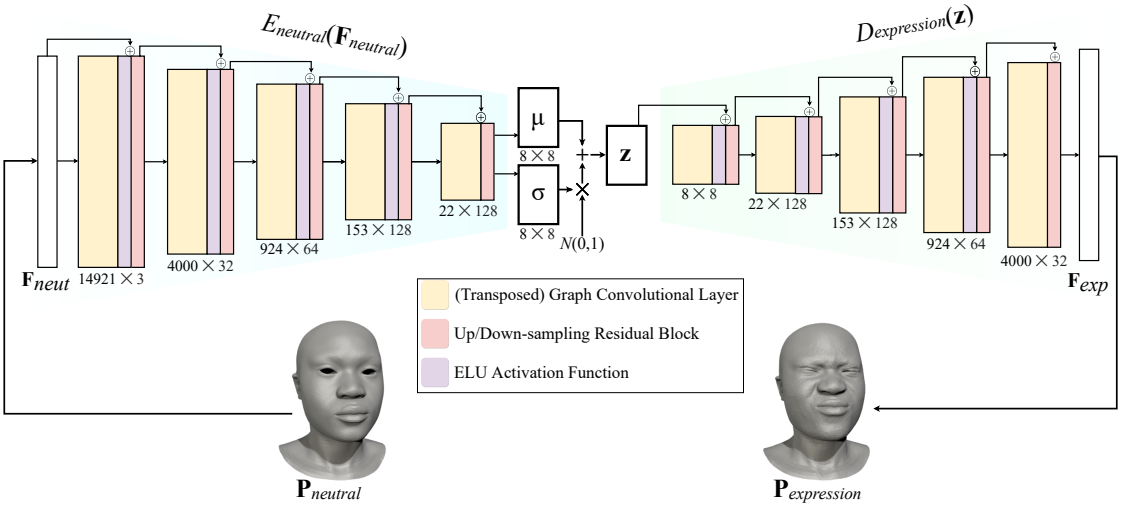


Figure 5.1 Diagram of the proposed end-to-end approach to generate personalised blendshapes.

using differential representation improves perceptual and geometric quality of generated personalised blendshapes.

Two personalised blendshapes generation methods are described in Section 5.2. The proposed methods use graph variational autoencoders and each of them is implemented in two variants: with standardised Euclidean coordinates representation and with the normalised deformation representation. Section 5.2.1 covers an end-to-end approach, which is then extended to a two-step approach in Section 5.2.2. Subsequently, both approaches with different representation variants and the Deformation Transfer method are compared in Section 5.2.3. Next, Section 5.2 covers the proposed Personalised Blendshapes Generation with Spectral Mesh Processing (PBS) method. Comparisons of PBS to other methods are provided in Section 5.3.5.

5.2 Personalised Blendshapes Generation using Variational Graph Autoencoders

5.2.1 End-to-end approach

It is proposed to employ graph autoencoder architecture to model a mapping between a face in a neutral pose $\mathbf{P}_{\text{neutral}}$ and its corresponding expression $\mathbf{P}_{\text{expression}}$. Figure 5.1 provides an overview of this approach. Neutral face meshes $\mathbf{P}_{\text{neutral}}$ are transformed to $\mathbf{F}_{\text{neut}}$, while expression meshes $\mathbf{P}_{\text{expression}}$ are transformed to $\mathbf{F}_{\text{neut}}$. These transformations, which are denoted $g(\cdot)$, involve preprocessing steps, which were detailed in Section 3.3.8.

Neutral face encoder $E_{neutral}$ encodes $\mathbf{F}_{\text{neut}}$ to latent mean μ and deviation σ. Mean and deviation are used to sample latent codes $\mathbf{Z}_{exp}$ from the distribution $\mathcal{N}(\mu,\sigma)$. While in the reconstruction task $\mathbf{Z}_{exp}$ represents a learned distribution of neutral meshes, here $\mathbf{Z}_{exp}$ is a learned representation of only those neutral face features which determine variation in performing a given expression.

Neutral and expression representation

Given a dataset of triangle neutral pose meshes with shared connectivity and their corresponding sets of expressions, neutral face meshes are translated so that centroid of each neutral face mesh is at the origin. Each corresponding expression is translated by the same vector. Subsequently, mean of neutral meshes is computed and denoted as $\bar{\mathbf{P}}_{neutral}$. Neutral meshes are rigidly registered to the mean and corresponding expressions are transformed by the same rigid transformation. Afterwards, all shapes are uniformly scaled by the same scalar, such that all meshes are in a cube whose sides are two units long. Mean of the resulting neutral face meshes is denoted as $\bar{\mathbf{P}}_{neutral}$, and means of identities performing n-th expression are denoted as $\bar{\mathbf{P}}_{expression\ n}$, where $n \in \{1,...,N\}$ is an index of an expression.

$$\hat{\mathbf{P}}_{neutral} = \mathbf{P}_{neutral} - \bar{\mathbf{P}}_{neutral}, \tag{5.1}$$

$$\mathbf{F}_{neutral} = g(\hat{\mathbf{P}}_{neutral}) \tag{5.2}$$

where $g(\cdot)$ is a function which transforms its input onto arbitrary representation, for example Euclidean coordinates (Euclidean), standardised Euclidean coordinates (Euclidean Std.), deformation representation (DR) or normalised deformation representation (DR Norm.).

$$\hat{\mathbf{P}}_{expression\ n} = \mathbf{P}_{expression\ n} - \mathbf{P}_{neutral}, \tag{5.3}$$

$$\mathbf{F}_{expression\ n} = g(\hat{\mathbf{P}}_{expression\ n}) \tag{5.4}$$

Neural network and training

The network is based on a fully convolutional variational autoencoder. The encoder $E_{neutral}$ and decoder $D_{expression}$ use convolution/ transposed convolution operations and upsampling/downsampling residual layers introduced in [151]. The encoder consists of 5 graph convolutional layers with stride = 2, kernel radius = 2, basis weights = 35, and channel dimensions = [$|\mathbf{f}|$, 32, 62, 128, 128, 8]. All of them, except the last one, are followed by *ELU* [24] activation function. The outputs from *ELU* are added to outputs from a downsampling residual layer. The decoders mirror encoders with 5 blocks of graph transposed convolutional layers followed by *ELU*s and upsampling residual layers.

Learnable parameters of the model are iteratively updated through backpropagation with Adam [68] optimiser in terms of loss

$$L = ||\mathsf{denorm}(\mathbf{F}_{exp}) - \mathsf{denorm}(D_{expression}(\mathbf{Z}))||_1 + \phi \mathsf{KL}(\mathcal{N}(0,1)||p(\mathbf{Z}|\mathbf{F}_{neut})), \tag{5.5}$$

when $g(\cdot)$ transforms into normalised representation, and

$$L = ||\mathsf{destd}(\mathbf{F}_{exp}) - (D_{expression}(\mathbf{Z}))||_1 + \phi \mathsf{KL}(\mathcal{N}(0,1)||p(\mathbf{Z}|\mathbf{F}_{neut})), \tag{5.6}$$

when $g(\cdot)$ transforms into standardised representation. The first term of each loss function is L_1 norm of a difference between ground truth expression and the network's output. The second term is weighted Kullback–Leibler (KL) divergence scaled with a scalar ϕ.

Inference and postprocessing

At inference, the network outputs $\mathbf{F}_{expression}$. The following equation processes it to obtain the predicted mesh $\mathbf{P}'_{expression\ n}$:

$$\mathbf{P}_{expression\ n} = g^{-1}(\mathbf{F}_{expression\ n}) + \mathbf{P}_{neutral} \tag{5.7}$$

where $g^{-1}(\cdot)$ is inverse transform of $g(\cdot)$.

5.2.2 Two-step approach

In this section, an alternative strategy is proposed. In an end-to-end approach, latent space $\mathbf{Z}_{exp}$ is a learned representation of only those neutral face features which determine variation in performing a given expression. In contrast, a two-step approach first learns the latent space $\mathbf{Z}_{neut}$ of neutral facial shapes, and then it models a mapping between neutral face features $\mathbf{Z}_{neut}$ and personalised expressions $\mathbf{F}_{expression}$. Figure 5.2 shows the difference between the end-to-end and the two-step approach.

Step 1 - neutral face representation learning

The first step aims at training an encoder E_{neut} which maps neutral face $\mathbf{F}_{neut}$ onto a small set of parameters $\mathbf{Z}_{neut}$ which form a latent space of neutral face features. To learn encoder E_{neut}, a Mesh Autoencoder [151] is used. In this work, it is proposed to compare two-stage approach using signal in Euclidean coordinates representation and in normalised deformation representation. Therefore, the deep 3D Morphable Models used to learn E_{neut} are the same as $(\mathbf{F}_{(3.15)})$ and $VAE_{Mesh}(\mathbf{F}_{(3.18)})$ from Section 3.5. After training, learnable

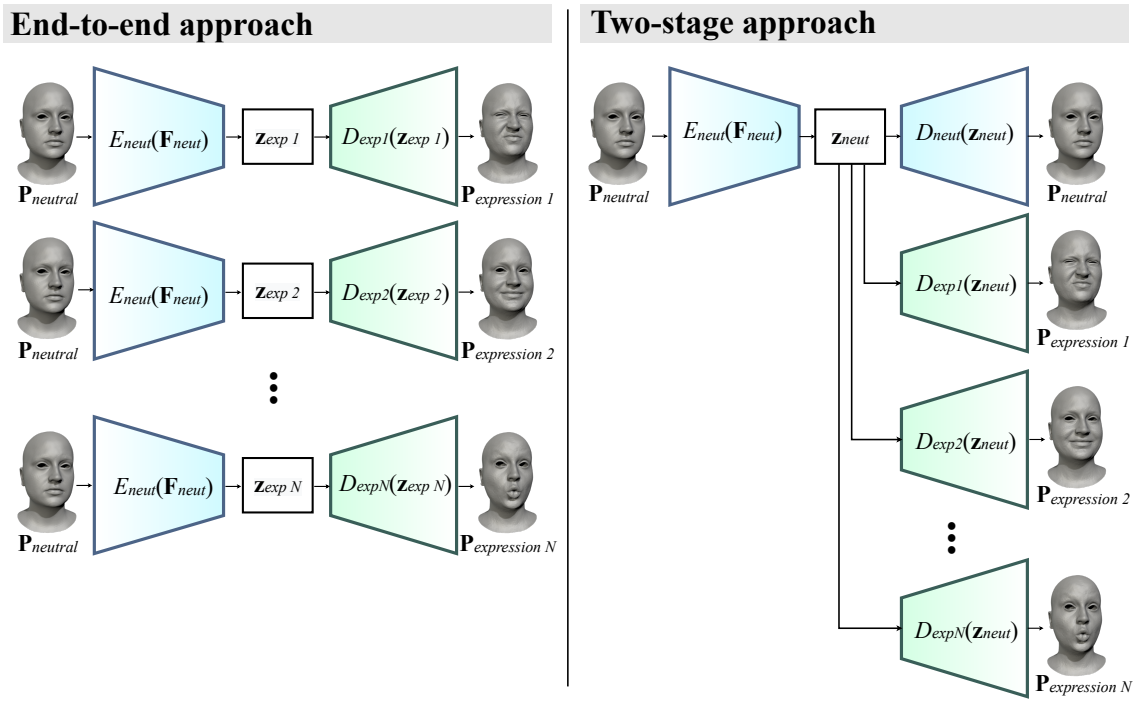


Figure 5.2 Comparison of an end-to-end training approach and the two-stage training strategy.

parameters of the resulting encoder E_{neut} are frozen. The decoder D_{neut} is discarded as it is used solely for the purpose of training E_{neut}.

Step 2 - decoding expressions

The second step aims at training a set of decoders $E_{exp\ n}$, where $n \in \{1,...,N\}$ is an index of an expression.

5.2.3 Comparative results

Comparative experiments are conducted on Facsimile [59] dataset, which consists of 202 identities, each performing a set of 19 expressions and a neutral pose. Experiments are conducted on three of these expressions: "face compression", "mouth wide" and "phoneme OO - brow raise - eyes open wide". Shapes are represented as triangle meshes, 14,921 vertices each. In experiments, the dataset is split into training, validation and test subsets in proportions 85:5:10. The networks are implemented in PyTorch 1.8 [38] and trained for 500 epochs with a learning rate of 10^4, batch size of 16 and a learning rate decay of 1% after each epoch. KL divergence weight $\phi = 10^{-6}$ and Adam optimiser's hyper-parameters are set to $\beta 1 = 0.9$ and $\beta 2 = 0.999$. All the experiments are trained and inferred on a single

NVIDIA GeForce GTX 1080 Ti with 16 GB memory. Data processing is performed on an Intel(R) Xeon(R) CPU E5-1630v4 running at 8 $\times$ 3.70 GHz using 32 GB RAM.

Two-step versus end-to-end approach

The end-to-end approach directly maps neutral faces to expressions in a single training process, offering higher accuracy since all layers are optimised together. It allows the model to automatically learn the most relevant facial features. This approach also generalises well to unseen faces if trained on diverse data. However, it requires all neutral identity shapes to be paired with expressions. This approach is computationally more expensive than the Two-step approach, making it less feasible for resource-constrained settings. Additionally, it can be harder to interpret, as the model directly learns the mapping.

In contrast, the two-step approach first trains an encoder to learn a latent representation of neutral faces and then maps this representation to facial expressions. This method allows for modular training, enabling the reuse of the encoder for other tasks and reducing redundancy. It also works with datasets where a subset of neutral identities is not paired with expressions. This way, neutral faces encoder can be pretrained on a larger dataset. Moreover, two-step approach provides better interpretability of a latent space. However, its accuracy is generally lower than the end-to-end approach because the two steps are optimized separately, leading to suboptimal feature learning.

The end-to-end approach is preferable when high accuracy is critical and sufficient data of paired neutral shapes and expressions is available. Meanwhile, the two-step approach is useful when a subset of neutrals is not paired with expressions, interpretability is needed, or computational resources are a constraint.

		End-to-end PB (Eucl. Std.)	Two-step PB (Eucl. Std.)		End-to-end PB (DR Norm.)	Two-step PB (DR Norm.)	
Face Compression	L1 Norm	7.034	6.985	↓ **1%**	9.662	10.239	↑ **6%**
	L2 Norm	3.632	3.674	↑ **1%**	5.640	6.018	↑ **7%**
	FMPD	26.289	42.73	↑ **63%**	10.170	15.923	↑ **57%**
	DAME	2.464	4.05	↑ **64%**	2.112	2.654	↑ **26%**
Mouth Wide	L1 Norm	5.436	5.298	↓ **3%**	6.556	7.053	↑ **8%**
	L2 Norm	2.355	2.314	↓ **2%**	2.848	2.915	↑ **2%**
	FMPD	25.519	34.781	↑ **36%**	8.591	14.037	↑ **63%**
	DAME	2.238	3.018	↑ **35%**	1.767	2.233	↑ **26%**
Phoneme OO Brow Raise Eyes Open Wide	L1 Norm	6.349	6.127	↓ **3%**	6.415	7.364	↑ **15%**
	L2 Norm	3.359	3.197	↓ **5%**	3.259	3.781	↑ **16%**
	FMPD	26.679	40.895	↑ **53%**	10.597	16.552	↑ **56%**
	DAME	2.460	3.728	↑ **52%**	2.074	2.636	↑ **27%**

Table 5.1 Quantitative comparison of spatial and perceptual error in two different training strategies: end-to-end and two-step, each using two different shape representations: standardised Euclidean coordinates (Eucl. Std.) and the normalised deformation representation (DR Norm.). Percentage points indicate change in error between the end-to-end and the two-step approaches.

Table 5.1 provides quantitative evaluation of an end-to-end PB and two-step PB coupled with standardised Euclidean coordinates and the normalised deformation representation. Both approaches are evaluated from geometric accuracy perspective using L_1 and L_2 norms between the predicted expressions and the ground truth expressions. They are also evaluated from perceptual perspective using DAME and FMPD error metrics.

The two-step PB with standardised Euclidean coordinates representation reduces geometric error by up to 3% of L_1 norm and up to 5% of L_2 norm comparing to end-to-end PB with standardised Euclidean coordinates. Nevertheless, these small improvements in geometric quality are overshadowed by substantial increase in perceptual FMPD and DAME metrics. Using end-to-end approach with standardised Euclidean coordinates representation leads to 63-64%, 35-36% and 52-53% higher perceptual loss in "face compression", "mouth wide" and "phoneme OO - brow raise - eyes open wide" expressions, respectively. Therefore, it can be concluded that end-to-end PB with standardised Euclidean coordinates representation should be preferred over its two-step counterpart due to significantly higher perceptual accuracy of synthesised meshes.

Comparison of the end-to-end and the two-step approaches which use the normalised deformation representation provides clear and unequivocal results. In this case, the end-to-end approach is better than the two-step approach from both, perceptual and geometric accuracy perspectives. Comparing to its end-to-end counterpart, the two-step PB with

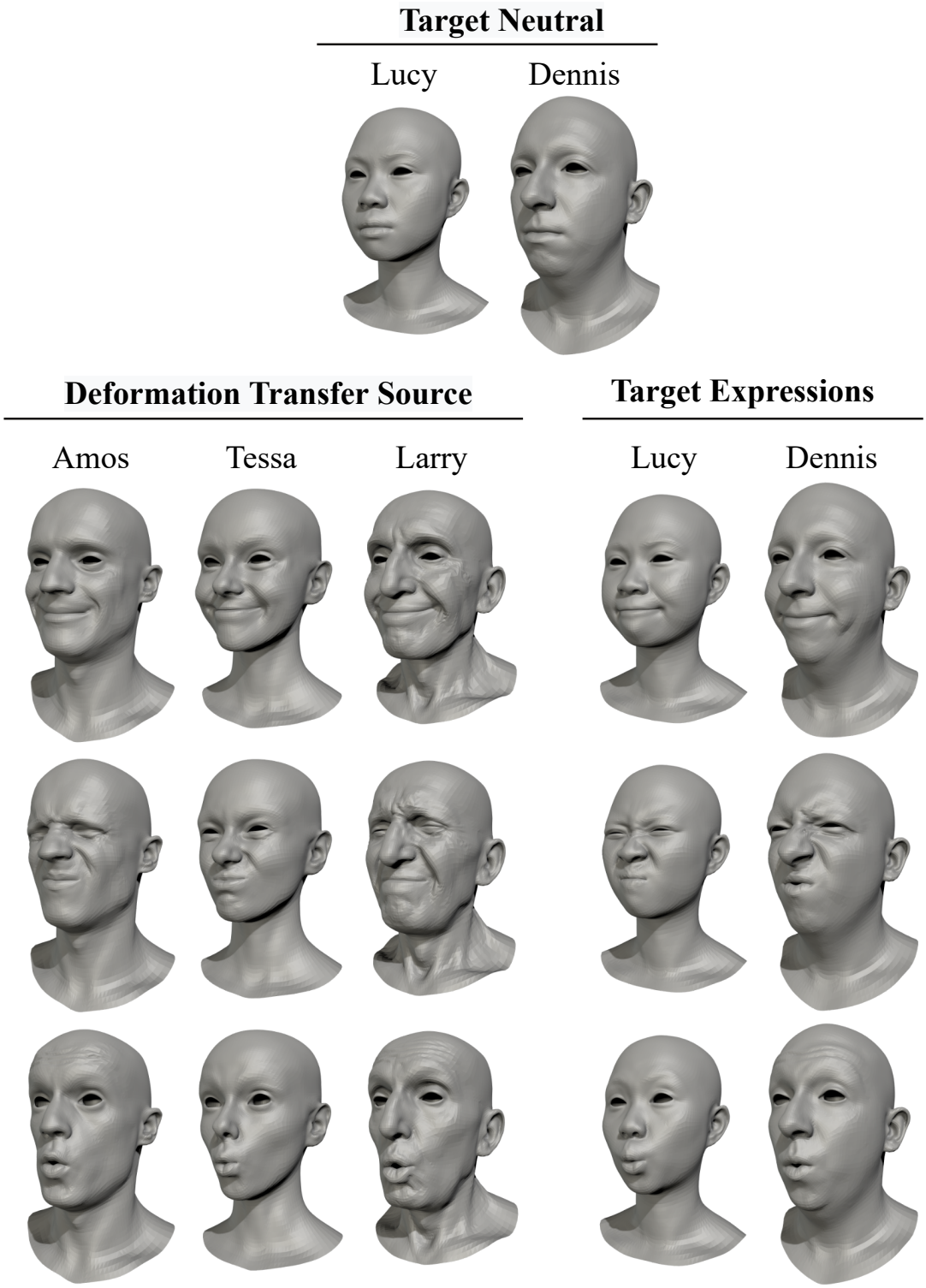


Figure 5.3 Visualisation of meshes used to generate results with the Deformation Transfer (DT) [120] method. Three different identities from the Facsimile [59] dataset with varying gender and age were used as sources of deformations (bottom left). Two target identities with neutral expression are selected for demonstration. A full validation dataset of identities is used in evaluation. The DT is used to transfer expressions from source shapes onto target neutral identities (top). The results are evaluated against the expected ground truth expressions (bottom right).

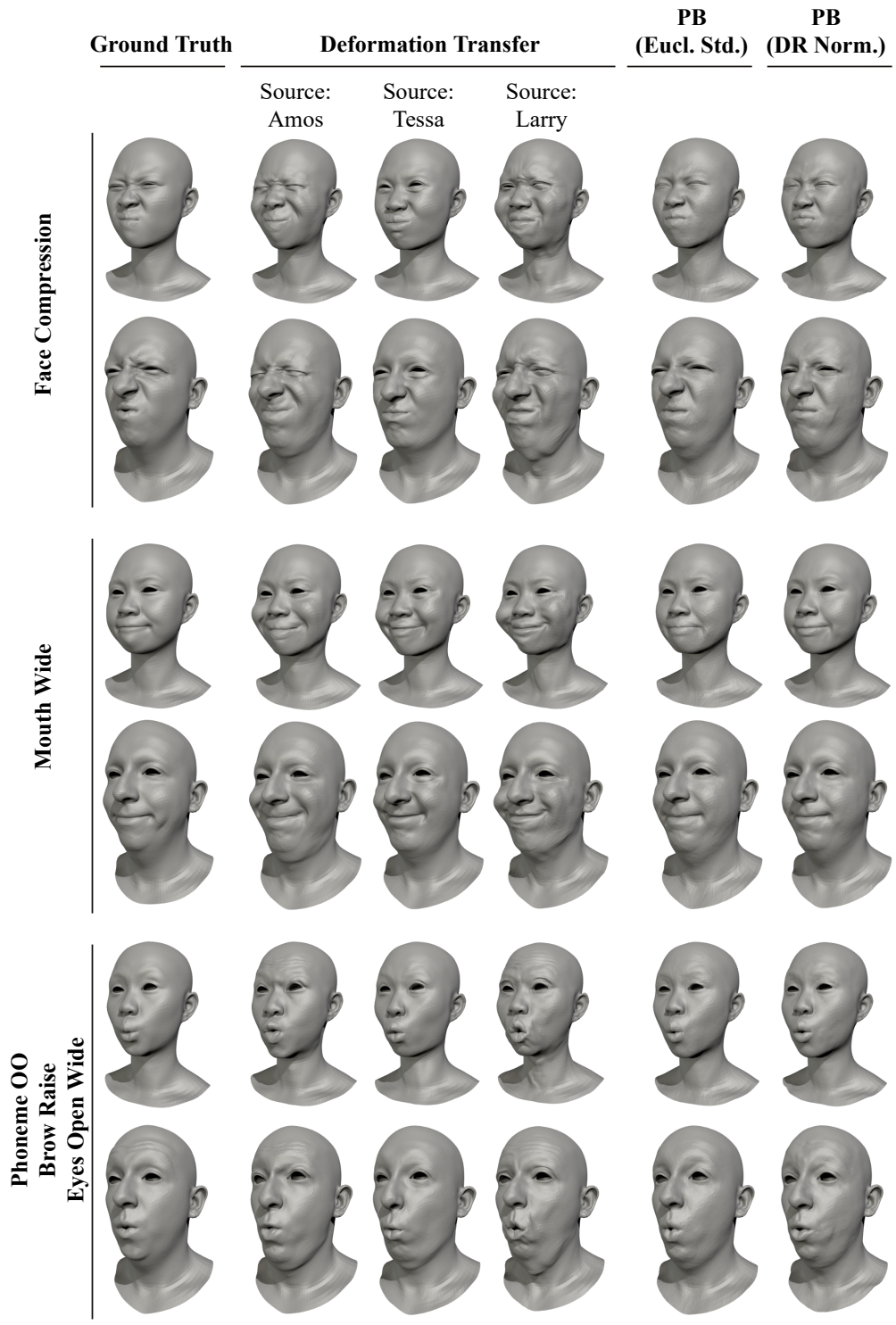


Figure 5.4 Visual comparison between expressions generated with three different methods: existing Deformation Transfer [120] method using three different sources of deformation, the proposed end-to-end approach with standardised Euclidean coordinates (Eucl. Std.) and the proposed end-to-end approach with normalised deformation representation (DR Norm.). It should be noted that the Deformation Transfer [120] method is compared here as it is a standard industrial approach for facial blendshapes generation. DT uses a single set of sources and its lower performance is expected when compared against models which are trained on the datasets of many identities.

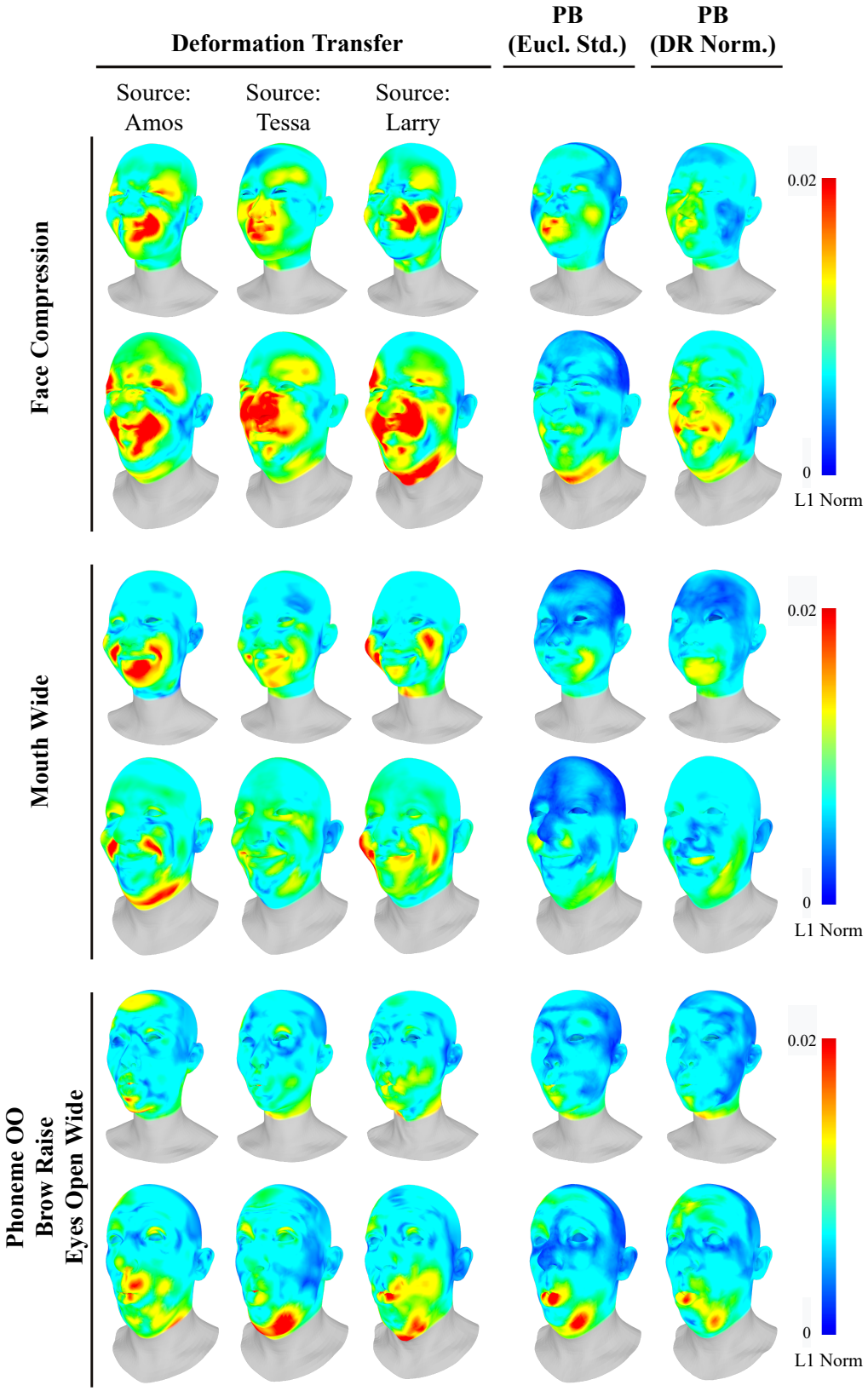


Figure 5.5 Qualitative comparison of synthesised expressions from Facsimile test dataset. Expressions generated by our proposed methods achieve lower L_1 error compared to the results from the Deformation Transfer.

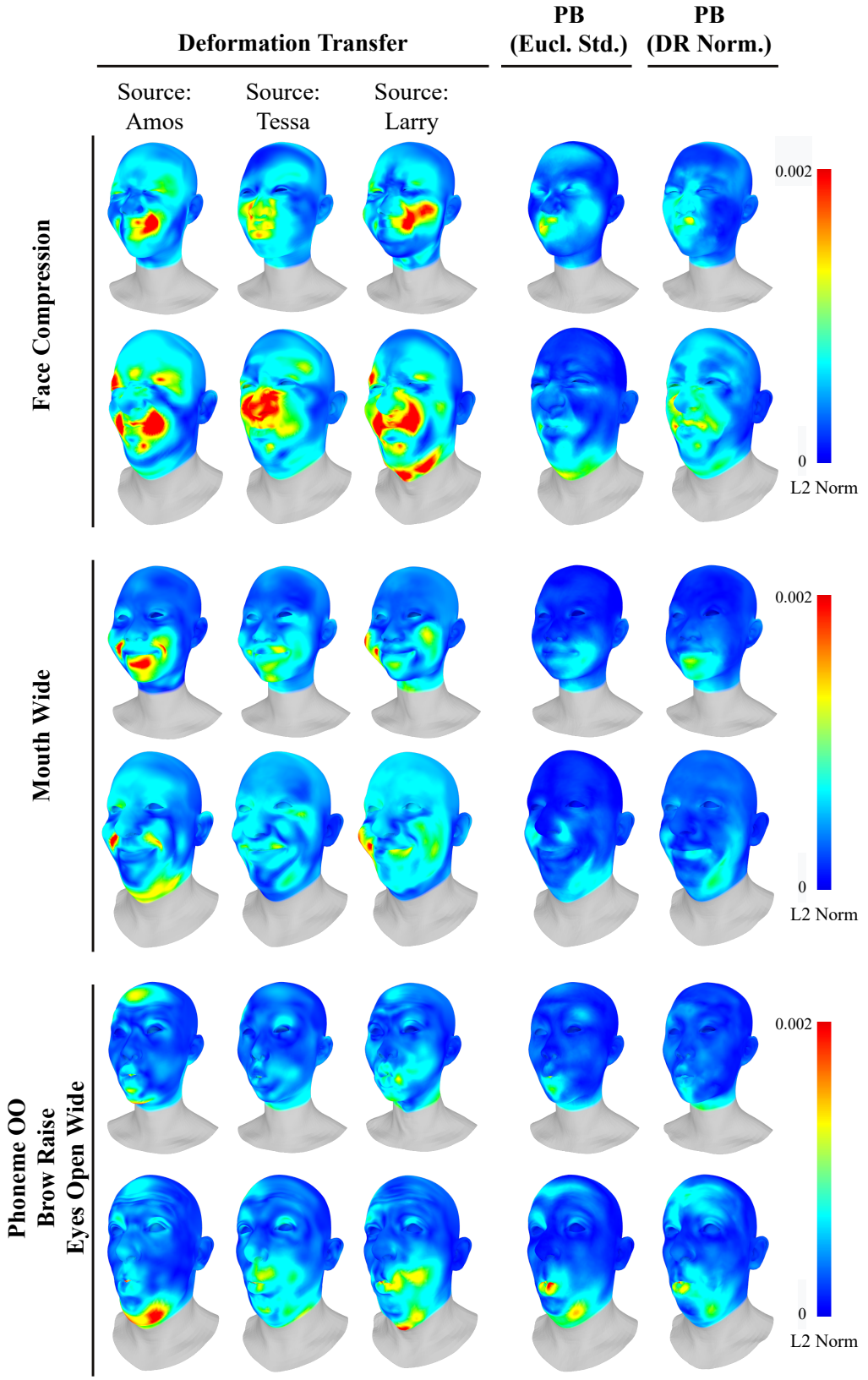


Figure 5.6 Qualitative comparison of synthesised expressions from Facsimile test dataset. Expressions generated by our proposed methods achieve lower L_2 error compared to the results from the Deformation Transfer.

the normalised deformation representation results in 6%, 8% and 15% higher L_1 norm error and 7%, 2% and 16% higher L_2 norm error in "face compression", "mouth wide" and "phoneme OO - brow raise - eyes open wide" expressions, respectively. The two-step approach also leads to higher perceptual error with FMPD increase by 56-63% and DAME increase by 26-27% in comparison to end-to-end PB with the normalised deformation representation. The results clearly demonstrate that end-to-end PB with the normalised deformation representation should be preferred over its two-step counterpart.

End-to-end PB versus Deformation Transfer

The proposed end-to-end PB networks using two variants of shape representations (standardised Euclidean coordinates and the normalised deformation representation) are compared with the existing Deformation Transfer [120] (DT) method. Figure 5.3 presents the identities and expressions used as sources in the DT process. It should be noted that the Deformation Transfer [120] method is compared here as it is a standard industrial approach for facial blendshapes generation. DT uses a single set of sources and its lower performance is expected when compared against models which are trained on the datasets of many identities. For better comparison, point-wise correspondence between source and target meshes is given.

Table 5.2 presents quantitative results which compare geometric L_1 and L_2 norm error, as well as perceptual FMPD and DAME. End-to-end PB outperforms the DT on L_1 and L_2 norm error across different expressions. Interestingly, in two cases the DT yields lower FMPD error than end-to-end PB. However, across all three expressions, it achieves higher DAME when compared against end-to-end PB with DR norm. Unfortunately, the representation used in end-to-end PB significantly affects its performance on geometric and perceptual metrics. The benefits of using either representation are mutually exclusive. Therefore, this problem is addressed in Section 5.3 which proposes use of spectral mesh processing to utilise benefits of both of these representations simultaneously.

Figures 5.4 - 5.6 allow for qualitative comparison between the methods. Expressions generated with the Deformation Transfer vary depending on the source of deformation. Distinct features of source identities can be identified on output expressions.

		Deformation Transfer [120]	End-to-end PB (Eucl. Std.)	End-to-end PB (DR Norm.)
Face Compression	L1 Norm	11.324	**7.034**	9.662
	L2 Norm	7.611	**3.632**	5.640
	FMPD	**8.028**	26.289	10.170
	DAME	2.228	2.464	**2.112**
Mouth Wide	L1 Norm	8.679	**5.436**	6.556
	L2 Norm	4.619	**2.355**	2.848
	FMPD	8.960	25.519	**8.591**
	DAME	1.997	2.238	**1.767**
Phoneme OO Brow Raise Eyes Open Wide	L1 Norm	8.400	**6.349**	6.415
	L2 Norm	5.076	3.359	**3.259**
	FMPD	**9.077**	26.679	10.597
	DAME	2.292	2.460	**2.074**

Table 5.2 Quantitative comparison of spatial and perceptual discrepancy between ground truth expressions and the expressions generated with three methods: existing Deformation Transfer [120] method, the proposed end-to-end approach with standardised Euclidean coordinates (Eucl. Std.) and the proposed end-to-end approach with normalised deformation representation (DR Norm.).

5.3 Personalised Blendshapes Generation with Spectral Mesh Processing

5.3.1 Method overview

The proposed method is outlined in Figure 5.7. In the first step, preprocessed meshes $\mathbf{P}_{neutral}$ are transformed to standardised Euclidean vectors $\mathbf{F}_{neut.spatial}$ and normalised deformation representation (DR Norm.) $\mathbf{F}_{neut.diff}$. Analogously, preprocessed meshes $\mathbf{P}_{expression}$ are transformed to $\mathbf{F}_{expr.spatial}$ and $\mathbf{F}_{neut.diff}$.

In the second step, two neural networks are trained simultaneously. The first network models the mapping between neutral face identities and their corresponding expressions in standardised Euclidean coordinates representation, while the second network models this mapping in normalised deformation representation. The architecture of each of the networks is the same as the one described in Section 5.2.1.

In the third step, Euclidean coordinates $\mathbf{P}'_{expr.spatial}$ and $\mathbf{P}'_{expr.diff}$ are decoded from their respective outputs $\mathbf{F}'_{expr.spatial}$ and $\mathbf{F}'_{expr.diff}$. Both $\mathbf{P}'_{expr.spatial}$ and $\mathbf{P}'_{expr.diff}$ are predictions

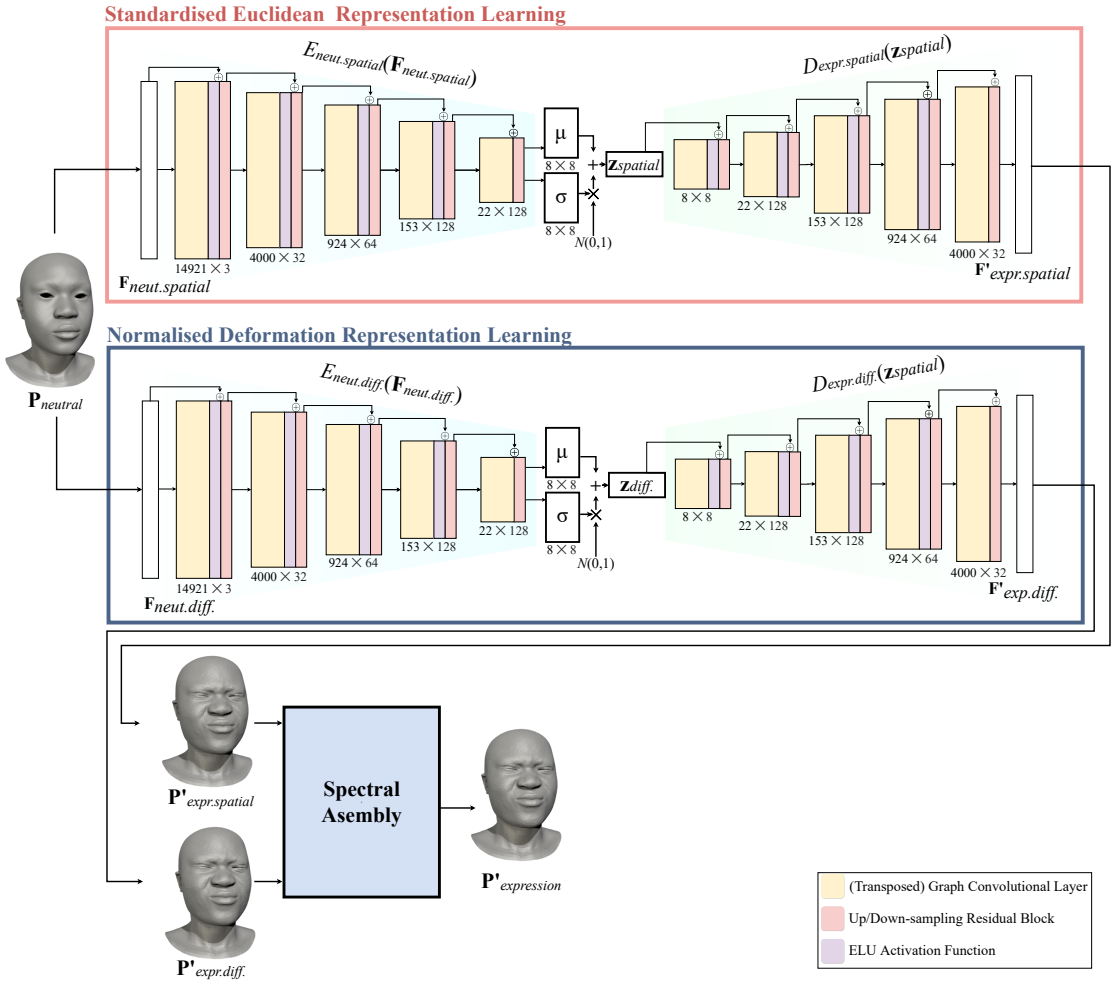


Figure 5.7 Overview of the proposed Personalised Blendshapes Generation with Spectral Mesh Processing approach.

of an expression $\mathbf{P}_{expression}$. However, it is expected that $\mathbf{P}'_{expr.spatial}$ has higher spatial fidelity and $\mathbf{P}'_{expr.diff}$ has higher perceptual fidelity. Therefore, both meshes are combined together to obtain the final vertex positions $\mathbf{P}'_{expression}$.

5.3.2 Data preprocessing and network training

Inputs $\mathbf{F}_{neut.spatial}$, $\mathbf{F}_{neut.diff}$ are calculated following Equations 5.1 and 5.2, whilst Equations 5.3 and 5.3 are used to calculate expected outputs $\mathbf{F}_{expr.spatial}$ and $\mathbf{P}'_{expr.diff}$. In case of spatial representation, function $g(\cdot)$ encodes to standardised Euclidean coordinates. For differential representation, function $g(\cdot)$ encodes to normalised deformation representation.

Learnable parameters of the model are iteratively updated through backpropagation with Adam [68] optimiser in terms of loss

$$L_{spatial} = ||\mathsf{denorm}(\mathbf{F}_{expr.spatial}) - \mathsf{denorm}(D_{neut.spatial}(\mathbf{Z}_{spatial}))||_1 + \phi \mathsf{KL}(\mathcal{N}(0,1)||p(\mathbf{Z}_{spatial}|\mathbf{F}_{neut.spatial})), \tag{5.8}$$

$$L_{differential} = ||\mathsf{destd}(\mathbf{F}_{expr.diff.}) - D_{expr.diff.}(\mathbf{Z}_{diff.})||_1 + \phi \mathsf{KL}(\mathcal{N}(0,1)||p(\mathbf{Z}_{diff.}|\mathbf{F}_{neut.diff.})). \tag{5.9}$$

5.3.3 Inference and spectral assembly

Signal $\mathbf{F}'_{expr.spatial}$ and $\mathbf{F}'_{expr.diff}$ is converted to vertex positions $\mathbf{P}'_{expr.spatial}$ and $\mathbf{P}'_{expr.diff}$. Consequently, $\mathbf{P}'_{expr.spatial}$ and $\mathbf{P}'_{expr.diff}$ are combined in the following way:

$$\mathbf{P}'_{expression} = (\mathbf{I} - \mathbf{X})\mathbf{P}'_{expr.diff.} + \mathbf{X}\mathbf{P}'_{expr.spatial}, \tag{5.10}$$

where $\mathbf{X}$ is the matrix from Equation (4.11).

5.3.4 Pareto-optimal partitions

This section explores the impact of the parameter k on spatial and perceptual fidelity of synthesised expressions. It can be hypothesised that for each expression there exists a different Pareto front of optimal values of k.

To test this hypothesis, Personalised Blendshapes Generation with Spectral Mesh Processing networks are trained on Facsimile training set to generate three expressions: "face compression", "mouth wide" and "phoneme OO - brow raise - eyes open wide". Subsequently, the networks are inferred with a validation Facsimile dataset and spectral assembly is performed using different parameters k, such that $k \in \mathscr{K} = \{0, 10, 20, ..., 90, 100, 120, 140, .., 180, 200, 230, 260, ..., 470, 500, 550, 600, ..., 950, 1000, 14921\}, |\mathscr{K}| = 37$. Here, $k = 0 \implies \mathbf{P}'_{expression} = \mathbf{P}'_{expr.diff.}$ and $k = 14921 \implies \mathbf{P}'_{expression} = \mathbf{P}'_{expr.spatial}$.

Next, the personalised blendshapes synthesised using values from the set $|\mathscr{K}|$ are evaluated both perceptually and geometrically to form a 2-dimensional space of solutions. In Figure 5.8, each choice of the parameter k is plot against perceptual FMPD error and geometric L_1 norm error.

The results prove that Pareto-optimal solutions vary across expressions. It can be observed that, in principle, lower values of k yield lower perceptual error and higher geometric error. Higher values of k tend to result in higher perceptual error and, up to certain expression-specific threshold, lower geometric error.

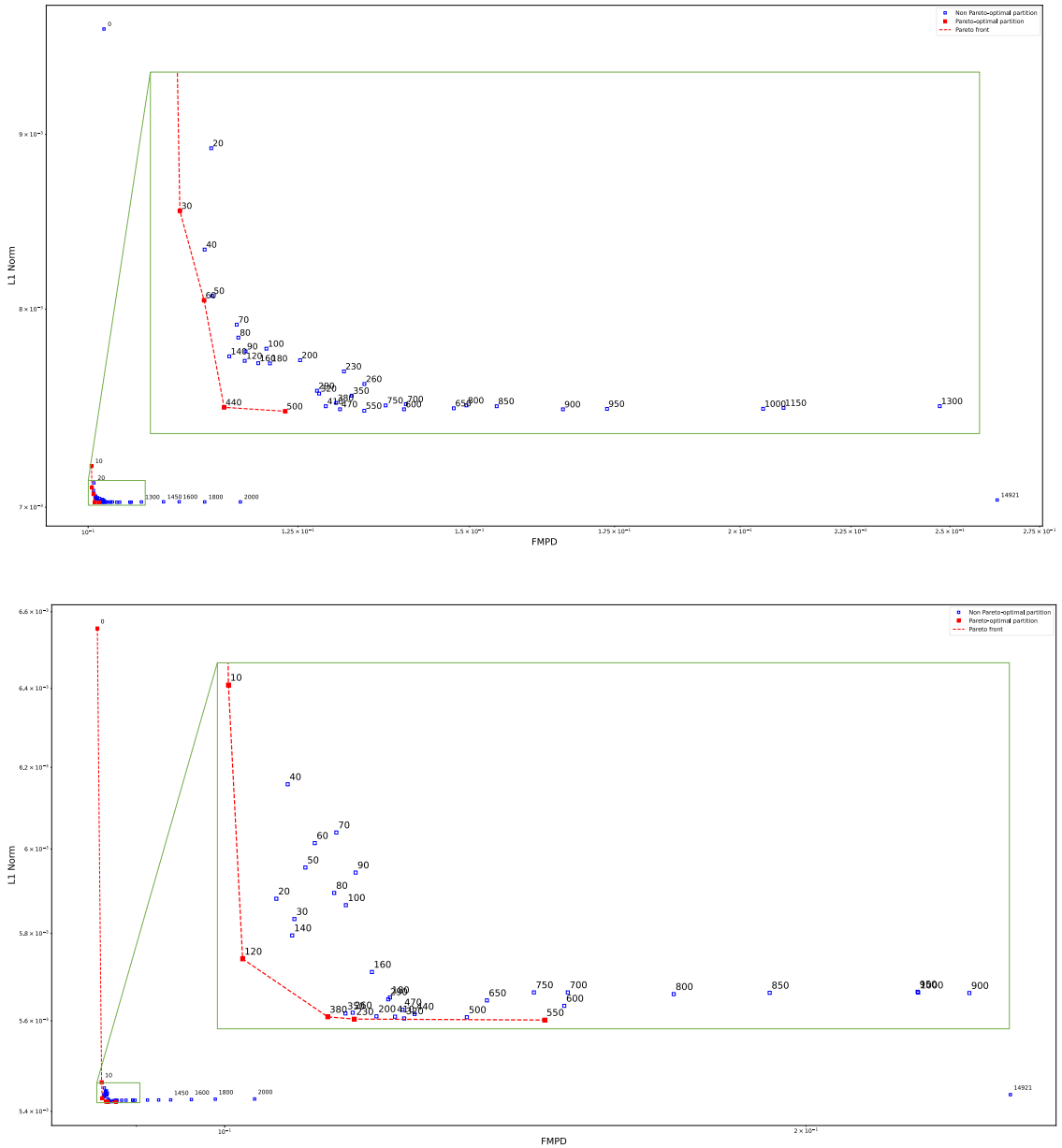


Figure 5.8 *Cont.*

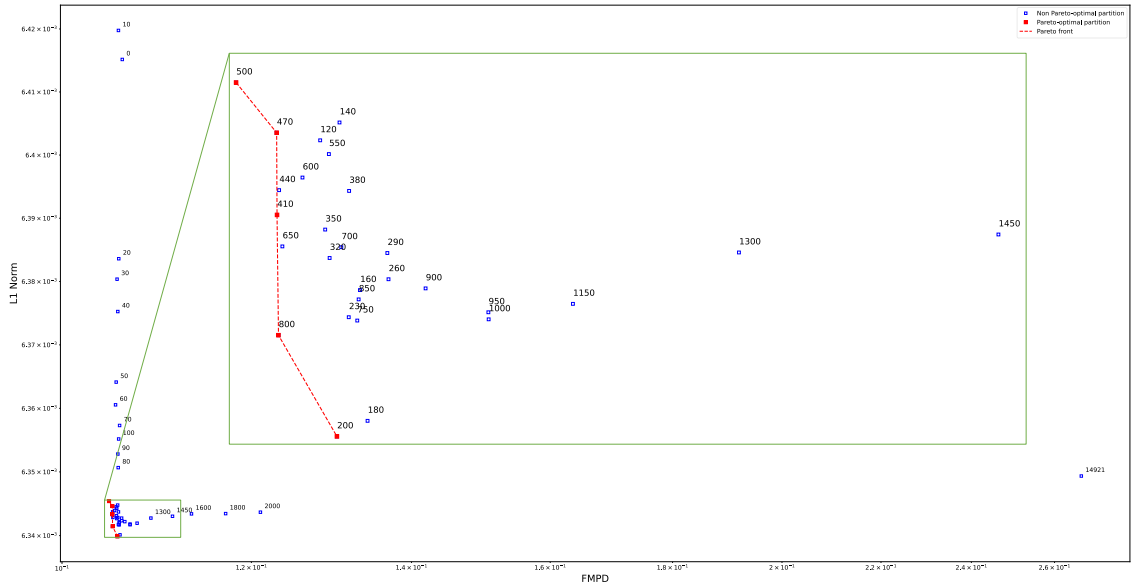


Figure 5.8 Comparison of the effect of different values of k (blue points) on perceptual and spatial error between the ground truth and the synthesised expressions. Comparison is performed using identities from validation Facsimile dataset.The plots depict results from spectral assembly of the following expressions: "face compression" (top), "mouth wide" (middle) and "phoneme OO - brow raise - eyes open wide" (bottom). The parameter k affects the trade-off between the perceptual error measured with FMPD and spatial fidelity measured with L_1 norm. Values of k which form a Pareto front of optimal solutions when considering solely perceptual quality and spatial fidelity are connected with a red line.

Interestingly, values $k = 0$ and $k = 14921$ are not always part of the Pareto front. It means that in most evaluated cases the proposed method is capable of outperforming an end-to-end PB with DR Norm. approach ($k = 0$) on perceptual FMPD metric and in all compared cases it improves upon an end-to-end PB with Euclidean Std. ($k = 14921$) when considering the spatial L_1 norm metric.

For "face compression" Facsimile expression, the following k parameters are Pareto-optimal: 10, 30, 440 and 500. Based on the knee of a curve, $k = 440$ is selected as the solution which balances both perceptual and spatial fidelity objectives. In the case of "mouth wide" Facsimile expression, the following k parameters are Pareto-optimal: 0, 10, 120, 380, 230 and 550. The knee of a curve suggests $k = 380$ as a well-balanced choice. Finally, for "phoneme OO - brow raise - eyes open wide" Facsimile expression, the Pareto front contains the following values of k: 500, 470, 800 and 200. As the slope of the Pareto front is steep, the value of $k = 200$ is considered a balanced choice.

5.3.5 Comparative results

Figure 5.9 compares different blendshapes synthesis methods in terms of geometric L_1 norm error and the perceptual DAME and FMPD errors. The following methods are compared: the proposed Personalised Blendshapes Generation with Spectral Mesh Processing (PBS) from Section 5.3, the end-to-end PB with DR Norm. and the end-to-end PB with Euclidean Std. methods from Section 5.2.1, and the Deformation Transfer [120]. The methods are evaluated on three different expressions. Pareto-optimal values of parameter k were used for each expression in the proposed PBS method, as described in Section 5.3.4.

Clearly, the Deformation Transfer (DT) produces least accurate blendshapes from geometric perspective. This result is expected, as the method uses a single set of source shapes and it is compared against a model which learns from examples of many identities. DT yields the highest L_1 norm error compared to all other methods across all three expressions. DT results in 61.11%, 60.15% and 32.47% higher L_1 norm error than the proposed PBS method. From perceptual perspective, the results are more ambiguous. On DAME metric, DT yields higher error than the proposed PBS and end-to-end PB with DR Norm. and it is only better than end-to-end PB with Euclidean Std. On the other hand, on FMPD metric, DT gives the lowest error of all the compared methods on two of three expressions.

The end-to-end PB with Euclidean Std. yields least favourable results, with the highest perceptual error of all the methods and lack of geometric accuracy gains when compared to the proposed PBS method. It gives 1.61, 1.94 and 1.53 times higher FMPD and 0.14%, 0.37% and 016% higher L_1 norm error compared to the proposed PBS method.

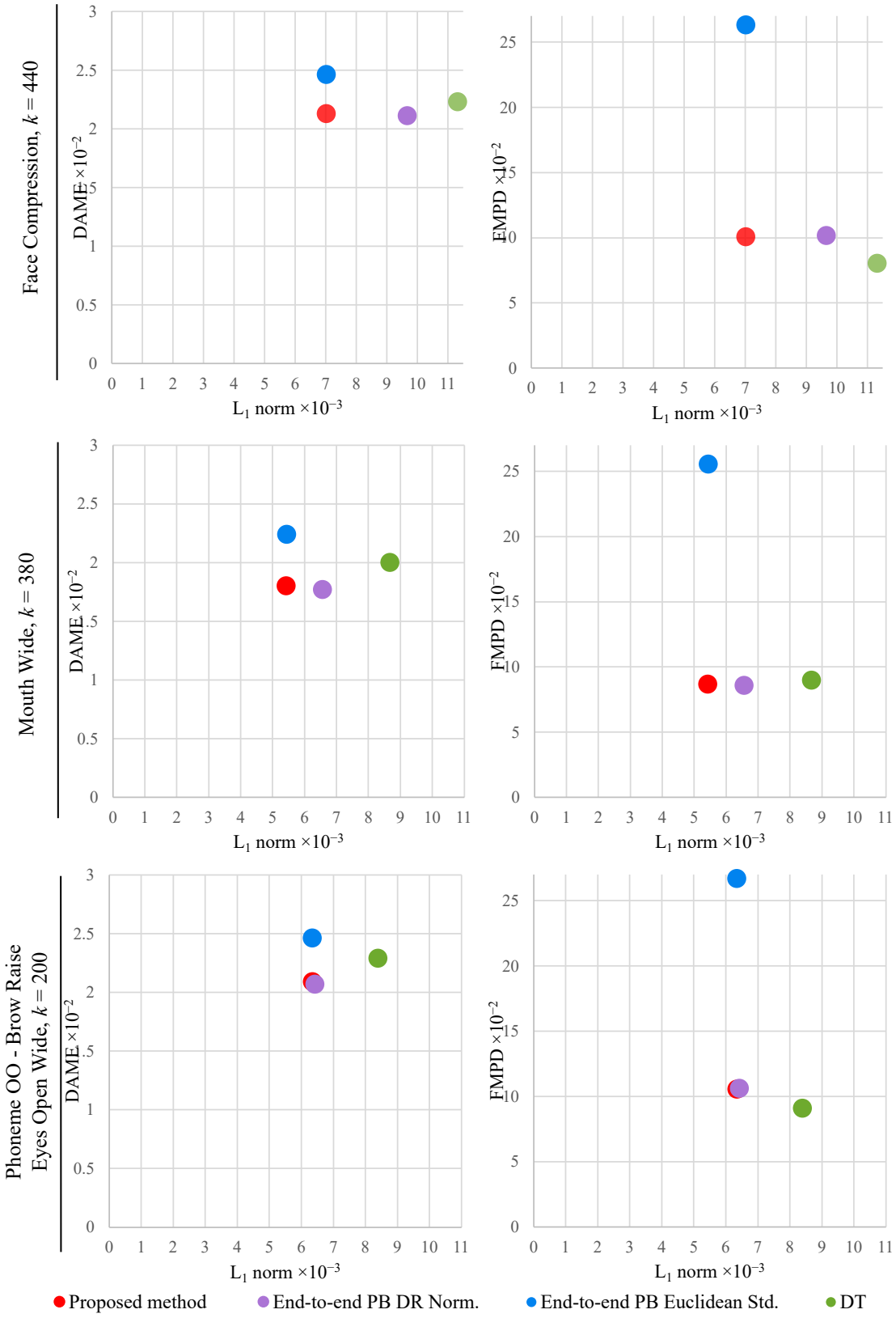


Figure 5.9 Comparison of different blendshapes generation methods in terms of perceptual DAME and spatial L_1 error between the expected ground truth expression and the expression synthesised with each method. Methods are evaluated on a test Facsimile dataset.

The end-to-end PB with DR Norm. consistently underperforms on L_1 norm metric compared to PBS. Despite small improvements on DAME perceptual metric, where end-to-end PB with DR Norm. yields just , it results in higher FMPD than PBS. Interestingly, its results are comparable to PBS both perceptually and geometrically in "phoneme OO - brow raise - eyes open wide" Facsimile expression. The reason for such a small difference between these results is a small spread of just 0.08 between L_1 norm error of End-to-end PB Euclidean Std. and End-to-end PB DR Norm.

Comparing to other methods, expressions synthesised with PBS have the lowest overall perceptual and geometric error. This is because the method utilises advantages of a global standardised Euclidean coordinates representation, and differential normalised deformation representation. Consequently, PBS yields similar perceptual error to end-to-end PB with DR Norm. and similar geometric error to end-to-end PB with Euclidean Std., thus providing the best overall results.

5.4 Conclusions

In the first part of this chapter, an end-to-end and two-step approaches have been proposed to train a graph variational autoencoder architecture which models a mapping between a face in a neutral pose and its corresponding expression. Quantitative and qualitative evaluation have proven hypothesis 1 from Section 5.1. It was demonstrated that graph variational autoencoders can improve upon the perceptual and geometric accuracy of expressions generated with the deformation transfer (DT) [120] method, which is current industrial standard practice. However, it needs to be noted that DT uses a single set of source shapes, while graph variational autoencoders learn from face deformations across many identities.

Experimental results presented in this chapter have shown that the end-to-end approach outperforms the two-step approach. Despite minor improvements of 1-5% in L_1 and L_2 norm error, the two-step PB with standardised Euclidean coordinates representation yields significantly higher perceptual error comparing to end-to-end PB with standardised Euclidean coordinates. Regarding the two-step PB with normalised deformation representation, its results yield higher error across all metrics when compared to the end-to-end PB with normalised deformation representation. Therefore, the hypothesis 2 from Section 5.1 has been disproven.

In the second part of this chapter, Personalised Blendshapes Generation with Spectral Mesh Processing (PBS) method has been proposed to improve perceptual and geometric quality of synthesised expressions. The method built upon findings about global and differential shape representations in deep 3D morphable models from Chapter 3, the

proposed Deep Spectral Meshes method from Chapter 4 and the conclusions from the first part of this chapter. The proposed PBS method improved overall perceptual and geometric quality of synthesised expressions, which has proven the hypothesis 3 from Section 5.1.

This page is intentionally left blank.

CHAPTER6

CONCLUSIONS AND FUTURE WORK

6.1 Summary and conclusions

In response to industrial requirements, this thesis has addressed the challenges posed in the following research question: "How to generate and edit geometrically and perceptually accurate digital faces and personalised blendshapes?". Several hypotheses have been proposed to answer this question. This section covers the main conclusions.

Chapter 3 tested the hypothesis that using different input and output representations to deep 3D morphable models can improve upon existing mesh reconstruction methods in terms of either perceptual or geometric accuracy. To test this hypothesis, the Deep3DMM Comparison Platform has been designed to compare different deep 3D morphable models within a single framework. Configurations of five Deep3DMMs with four input and output representations were trained with three different datasets. The 60 experimental models were evaluated from geometric accuracy perspective using L_1 and L_2 norm metrics and from a perceptual quality perspective using DAME and FMPD metrics. Qualitative and quantitative analysis has demonstrated that the hypothesis is true. The proposed use of standardised Euclidean coordinates representation improved the geometric and perceptual quality of the Mesh Autoencoder [109] method, which originally used the Euclidean coordinates. Additionally, the proposed use of the DR improved the perceptual quality of all the compared methods on most datasets.

It has been hypothesised that input and output representations, which explicitly encode the surface properties, improve the perceptual quality of the resulting meshes, while those which explicitly encode the vertex positions in 3D space, improve the geometric accuracy of generated meshes. This hypothesis has been proven to be true using Deep3DMM Comparison Platform. It has been demonstrated that using Euclidean coordinates-based representations outperforms differential coordinates-based representations in geometric

accuracy, while differential coordinates-based representations achieve better results on perceptual DAME and FMPD metrics.

Chapter 4 tested the hypothesis that by integrating the representation which explicitly encodes the surface properties, and by integrating spectral mesh processing for decomposition of mesh displacements, geometric deep learning, and parametric models with graph neural networks, a new 3D facial mesh synthesis model can be developed, such that it would expose user parameters to control disentangled low- and high-frequency displacements, generate plausible facial shapes, and allow the user to control displacements independently at low- and high-frequency levels.

To test this hypothesis, spectral meshes were introduced to decompose mesh displacements into low- and high-frequency parts. Mesh partitioning requirements helped to identify the spectral mesh processing as the preferred method. Subsequently, these parts were represented with standardised Euclidean coordinates and the normalised deformation representation, respectively. Graph neural networks were proposed to reconstruct the inputs, which were later converted back to Euclidean coordinates to obtain the reconstructed 3D models. A Conditioning Factor was introduced to control the level of mutual conditioning of displacements at different frequencies.

Extensive experiments were conducted to validate and compare the proposed approach with previously published methods, both quantitatively and qualitatively. It was demonstrated that spectral decomposition of meshes without the mass matrix normalisation produces partitions with spatial frequency imbalance, because such partitioning is tessellation-dependent. On the other hand, it was shown that mass matrix normalisation counters this imbalance. Results demonstrated the capability of Deep Spectral Meshes approach to independently edit low- and high-frequency displacements of facial meshes. Moreover, the experiments illustrated the impact of the Conditioning Factor on balancing mutually exclusive objectives of independent control of displacements at different frequencies and generating plausible synthetic examples. The choice of parameter k has been investigated. It was shown that there exists a Pareto-front of optimal parameters k which satisfy joint objectives of perceptual and geometric accuracy of reconstructed meshes. As the parameter k increases, the perceptual error increases and geometric error decreases. Comparisons with existing methods demonstrate the superiority of this approach over Euclidean coordinates, standardised Euclidean coordinates, the normalised deformation representation (DR) and other methods, as assessed by both L_1 and perceptual metric evaluations. Consequently, the hypothesis has been proven to be true.

The significance of the proposed method is underscored through its applications presented in this thesis. The proposed method has been applied in mesh compression, enhancing the quality of the reconstructed meshes. Additionally, it has been employed in mesh interpolation, expanding the capability to generate a larger variety of new shapes

compared to direct interpolation between two facial meshes. This was achieved by independently interpolating both low-frequency and high-frequency parameters. Moreover, the proposed method has found application in multi-frequency editing, satisfying different editing requirements. It allows the alteration of the overall shape while preserving details by adjusting only high-frequency parameters. It also accommodates modifying fine details while maintaining the overall shape by editing only low-frequency parameters.

Finally, Chapter 5 tested the hypothesis that graph autoencoders can be used to model a mapping between the neutral face mesh and its personalised expression, such that it improves upon currently used Deformation Transfer method [95] from geometric and perceptual accuracy perspectives. This hypothesis has been proven to be true. Two different training approaches were compared. In an end-to-end approach, the model learns directly from neutral face mesh to output a personalised blendshape. In a two-stage approach, a parametric model of neutral faces is trained in the fist stage. In the second training stage, a graph model is trained to decode from features extracted from a neutral mesh to a personalised expression mesh. It has been demonstrated quantitatively that the end-to-end approach yields lower perceptual and geometric error and that both approaches outperform the Deformation Transfer.

Subsequently, work in Chapter 5 confirmed the hypothesis that spectral mesh processing applied to the proposed personalised expressions generation method can improve geometric and perceptual accuracy of synthesised expressions. A neural network inspired by Deep Spectral Meshes has been proposed to test this hypothesis.

6.2 Future work

Applications of Deep Spectral Meshes and spectral mesh processing introduced to geometric deep learning are potentially far-reaching, but several promising applications and areas of research which warrant further investigation have been identified:

- This work restricts its application to low- and high-frequency bands. The partition of mesh data into more than two frequency bands and an investigation into the relationship between the learning model and the frequency bands remain unexplored. The limitation to two frequency bands in the proposed approach is linked to the properties of the chosen mesh representations. Standardised Euclidean coordinates represent low-frequency information to ensure high point-wise accuracy of the generated meshes. The normalised deformation representation (DR) encodes high-frequency information for superior perceptual quality of the results. Future work on partitioning mesh data into more than two frequency bands could further explore the benefits of alternative representations at different frequency levels.

- Deep Spectral Meshes can be extended to address the problem of multi-frequency-based deformation transfer, which has not been investigated in existing research studies. The basic idea of multi-frequency-based deformation transfer involves decomposing source and target meshes into mean, low- and high-frequency parts. The differences between the source model and these parts at two different poses are determined and transferred to the corresponding bands of the target mesh. Subsequently, the graph neural network proposed in this paper can be employed to reconstruct a new shape for the target mesh with the pose of the source mesh.

- While the proposed approaches have been applied to deformable facial meshes, future work could extend the proposed methods to articulated shapes like hands and bodies. Existing research has proposed various methods to relate articulated shapes to their underlying skeleton. With this extension, articulated shapes can be decomposed into mean, low- and high-frequency parts. Then, the relationships between these parts and the movements of the skeleton of the articulated shapes can be investigated. These relationships can be used to synthesise mean, low- and high-frequency parts of new poses. The graph neural network proposed in this paper can then extract features, reconstruct them, and synthesise new shapes from the reconstructed features.

- The proposed Deep Spectral Meshes could be applied to the monocular 3D face reconstruction task to address the common challenge of balancing two conflicting effects: (1) regularisation of the parametric models, which accurately reconstructs global shape and (2) constraints from cues, which bring fine-level details. It can be theorised that independently regressing the coefficients representing high- and low-frequency information using different visual cues might help with balancing these conflicting effects.

- Deep Spectral Meshes could be extended to handle dynamic meshes. They key challenge in this extension is ensuring temporal coherence and maintaining computational efficiency at data preprocessing and inference stage.

Personalised expressions generation with PBS method can be extended in many research directions. The following potential avenues are suggested:

- The proposed PBS method uses neutral face mesh as input to synthesise a personalised expression. The method can be extended with multimodal guidance. Modalities which capture low-frequency information, such as pointclouds, could guide the graph variational autoencoder which uses standardised Euclidean coordinates representation. Modalities which capture higher-frequency information, such

as shape-from-shading, could guide the graph variational autoencoder which uses the normalised deformation representation.

- This work requires to train a separate neural network per expression. While this approach provides flexibility, which is often required in industrial setting, future work could explore training a conditional version, which allows to generate different expressions based on a given condition.

- In the proposed approach, assembly with spectral mesh processing is a postprocessing step. Therefore, comparing the influence of different parameters k is computationally inexpensive, as it does not require retraining of the networks. Nevertheless, future research could focus on developing a solution which would involve assembly with spectral mesh processing within a loss function, and the parameter k would be a learnable parameter. A modified, differentiable assembly with spectral mesh processing would need to be developed.

This page is intentionally left blank.

REFERENCES

[1] Abrevaya, V. F., Boukhayma, A., Wuhrer, S., and Boyer, E. (2019). A Decoupled 3D Facial Shape Model by Adversarial Training. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9419–9428.

[2] Agrawal, S., Pahuja, A., and Lucey, S. (2020). High accuracy face geometry capture using a smartphone video. In *Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020*, pages 81–90.

[3] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. *arXiv preprint arXiv:1701.07875*.

[4] Bagautdinov, T., Wu, C., Saragih, J., Fua, P., and Sheikh, Y. (2018). Modeling Facial Geometry Using Compositional VAEs. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3877–3886. IEEE.

[5] Bao, X., Tong, W., and Chen, F. (2023). A Spectral Segmentation Method for Large Meshes. *Communications in Mathematics and Statistics*, 11:583–607.

[6] Bao, Y., Ding, T., Huo, J., Liu, Y., Li, Y., Li, W., Gao, Y., and Luo, J. (2025). 3d gaussian splatting: Survey, technologies, challenges, and opportunities. *IEEE Transactions on Circuits and Systems for Video Technology*.

[7] Baran, I., Vlasic, D., Grinspun, E., and Popovi´c, J. P. (2009a). Semantic Deformation Transfer. *ACM SIGGRAPH 2009 papers*, pages 1–6.

[8] Baran, I., Vlasic, D., Grinspun, E., and Popovi´c, J. P. (2009b). Semantic Deformation Transfer. *ACM SIGGRAPH 2009 papers*, pages 1–6.

[9] Berthelot, D., Schumm, T., and Metz Google, L. (2017). BEGAN: Boundary Equilibrium Generative Adversarial Networks. *arXiv preprint arXiv:1703.10717*.

[10] Biewald, L. (2020). Experiment Tracking with Weights and Biases.

[11] Blanz, V. and Vetter, T. (1999). A Morphable Model For The Synthesis Of 3D Faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194.

[12] Booth, J., Roussos, A., Ververas, E., Antonakos, E., Ploumpis, S., Panagakis, Y., and Zafeiriou, S. (2018). 3D Reconstruction of 'In-the-Wild' Faces in Images and Videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(11):2638–2652.

[13] Booth, J., Roussos, A., Zafeiriou, S., Ponniah, A., and Dunaway, D. (2016). A 3D Morphable Model learnt from 10,000 faces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5543–5552.

[14] Bouritsas, G., Bokhnyak, S., Ploumpis, S., Zafeiriou, S., and Bronstein, M. (2019). Neural 3D morphable models: Spiral convolutional networks for 3D shape representation learning and generation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7212–7221.

[15] Bronstein, M. M., Bruna, J., Lecun, Y., Szlam, A., and Vandergheynst, P. (2017). Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42.

[16] Browatzki, B. and Wallraven, C. (2020). 3FabRec: Fast Few-shot Face alignment by Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6110–6120.

[17] Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2014). Spectral networks and deep locally connected networks on graphs. In *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, pages 1–14.

[18] Cao, C., Weng, Y., Zhou, S., Tong, Y., and Zhou, K. (2014). FaceWarehouse: A 3D facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):413–425.

[19] Chen, A., Chen, Z., Zhang, G., Mitchell, K., and Yu, J. (2019). Photo-Realistic Facial Details Synthesis From Single Image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9429–9439.

[20] Chen, G. and Wang, W. (2024). A survey on 3d gaussian splatting. *arXiv preprint arXiv:2401.03890*.

[21] Chen, Y., Wang, L., Li, Q., Xiao, H., Zhang, S., Yao, H., and Liu, Y. (2024). Monogaussianavatar: Monocular gaussian point-based head avatar. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–9.

[22] Chen, Z. and Kim, T. K. (2021). Learning feature aggregation for deep 3D morphable models. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 13159–13168.

[23] Cheng, S., Bronstein, M., Zhou, Y., Kotsia, I., Pantic, M., and Zafeiriou, S. (2019). MeshGAN: Non-linear 3D Morphable Models of Faces. *arXiv preprint arXiv:1903.10384*.

[24] Clevert, D. A., Unterthiner, T., and Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (ELUs). In *4th International Conference on Learning Representations, ICLR 2016*.

[25] Corsini, M., Larabi, M. C., Lavoué, G., Petřík, O., Váša, L., and Wang, K. (2013). Perceptual metrics for static and dynamic triangle meshes. *Computer Graphics Forum*, 32(1):101–125.

[26] Croitoru, F.-A., Hondru, V., Ionescu, R. T., and Shah, M. (2022). Diffusion Models in Vision: A Survey. 14(8):1–22.

[27] Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852.

[28] Deng, Y., Yang, J., Xu, S., Chen, D., Jia, Y., and Tong, X. (2019). Accurate 3D face reconstruction with weakly-supervised learning: From single image to image set. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 285–295.

[29] Denton, E., Chintala, S., Szlam, A., and Fergus, R. (2015). Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. *Advances in neural information processing systems*, pages 1486–1494.

[30] Dhamo, H., Nie, Y., Moreau, A., Song, J., Shaw, R., Zhou, Y., and Pérez-Pellitero, E. (2024). Headgas: Real-time animatable head avatars via 3d gaussian splatting. In *European Conference on Computer Vision*, pages 459–476. Springer.

[31] Dhariwal, P. and Nichol, A. (2021). Diffusion Models Beat GANs on Image Synthesis. *Advances in Neural Information Processing Systems*, 11:8780–8794.

[32] Dong, Q., Wang, Z., Li, M., Gao, J., Chen, S., Shu, Z., Xin, S., Tu, C., and W., W. (2023). Laplacian2Mesh: Laplacian-Based Mesh Understanding. *IEEE Transactions on Visualization &; Computer Graphics*, pages 1–13.

[33] Dubrovina, A. and Kimmel, R. (2010). Matching shapes by eigendecomposition of the Laplace-Beltrami operator. In *Proc. 3DPVT*, pages 1–8.

[34] Egger, B., Smith, W., Tewari, A., Wuhrer, S., Zollhoefer, M., Beeler, T., Bernard, F., Bolkart, T., Kortylewski, A., Romdhani, S., Theobalt, C., Blanz, V., and Vetter, T. (2020). 3D Morphable Face Models—Past, Present, and Future. *ACM Transactions on Graphics*, 39(5):1–38.

[35] Ekman, P., Friesen, W. V., and Hager, J. C. (2002). *Facial Action Coding System: The Manual*. Research Nexus division of Network Information Research Corporation, Salt Lake City, USA, 2nd edition.

[36] Engel, J., Koltun, V., and Cremers, D. (2018). Direct Sparse Odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625.

[37] Engel, J., Schöps, T., and Cremers, D. (2014). LSD-SLAM: Large-Scale Direct Monocular SLAM. *Computer Vision – ECCV*, 8690:834–849.

[38] Falcon, W. (2019). PyTorch Lightning.

[39] Feng, X. and Shi, M. (2009). Surface representation and processing. In *2009 8th IEEE International Conference on Cognitive Informatics*, pages 542–545. IEEE.

[40] Gain, J. and Bechmann, D. (2008). A survey of spatial deformation from a user-centered perspective. *ACM Transactions on Graphics*, 27(4).

[41] Gao, L., Lai, Y. K., Liang, D., Chen, S. Y., and Xia, S. (2016). Efficient and flexible deformation representation for data-driven surface modeling. *ACM Transactions on Graphics*, 35(5):158:1–158:17.

[42] Gao, L., Lai, Y.-K., Yang, J., Ling-Xiao, Z., Xia, S., and Kobbelt, L. (2019). Sparse Data Driven Mesh Deformation. In *IEEE Transactions on Visualization and Computer Graphics*, pages 1–15.

[43] Gao, Z., Yan, J., Zhai, G., Zhang, J., Yang, Y., and Yang, X. (2021). Learning Local Neighboring Structure for Robust 3D Shape Representation. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21) Learning*.

[44] Garland, M. and Heckbert, P. S. (1997). Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1997*, pages 209–216.

[45] Gecer, B., Lattas, A., Ploumpis, S., Deng, J., Papaioannou, A., Moschoglou, S., and Zafeiriou, S. (2019). Synthesizing Coupled 3D Face Modalities by Trunk-Branch Generative Adversarial Networks. *arXiv preprint arXiv:1909.02215*.

[46] Gong, S., Chen, L., Bronstein, M., and Zafeiriou, S. (2019). SpiralNet++: A fast and highly efficient mesh convolution operator. In *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*, pages 4141–4148.

[47] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Nets. *Advances in neural information processing systems*, pages 2672–2680.

[48] Gruber, A., Fratarcangeli, M., Zoss, G., Cattaneo, R., Beeler, T., Gross, M., and Bradley, D. (2020). Interactive Sculpting of Digital Faces Using an Anatomical Modeling Paradigm. *Eurographics Symposium on Geometry Processing*, 39(5).

[49] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved Training of Wasserstein GANs. *Advances in neural information processing systems*, pages 5767–5777.

[50] Hanocka, R., Fleishman, S., Hertz, A., Fish, N., Giryes, R., and Cohen, D. (2019). MeshCNN: A Network with an Edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12.

[51] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

[52] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:770–778.

[53] Hernandez, M., Hassner, T., Choi, J., and Medioni, G. (2017). Accurate 3D face reconstruction via prior constrained structure from motion. *Computers and Graphics (Pergamon)*, 66:14–22.

[54] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*.

[55] Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 2020-Decem(NeurIPS 2020):1–25.

[56] Hu, L., Zhang, H., Zhang, Y., Zhou, B., Liu, B., Zhang, S., and Nie, L. (2024). Gaussianavatar: Towards realistic human avatar modeling from a single video via animatable 3d gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 634–644.

[57] Huang, H., Li, Z., Sun, Z., and Tan, T. (2018). IntroVAE: Introspective Variational Autoencoders for Photographic Image Synthesis. In *Advances in neural information processing systems*, pages 52–63.

[58] Huang, X. and Belongie, S. (2017). Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-Octob, pages 1510–1519.

[59] Humain Limited (2022). Humain Limited - Research & Development.

[60] Ichim, A. E., Kadleček, P., Kavan, L., and Pauly, M. (2017). Phace: Physics-based Face Modeling and Animation. *ACM Trans. Graph*, 36(153):1–14.

[61] Jain, V. and Zhang, H. (2006). Robust 3D Shape Correspondence in the Spectral Domain. In *Proc. 3DPVT, Vol. 2*, pages 1–12.

[62] Jiang, Z. H., Wu, Q., Chen, K., and Zhang, J. (2019). Disentangled representation learning for 3D face shape. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 11949–11958. IEEE.

[63] Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2018). Progressive growing of GANs for improved quality, stability, and variation. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*.

[64] Karras, T., Laine, S., and Aila, T. (2019a). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June, pages 4396–4405.

[65] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2019b). Analyzing and Improving the Image Quality of StyleGAN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119.

[66] Kerbl, B., Kopanas, G., Leimkühler, T., and Drettakis, G. (2023). 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1.

[67] Kim, B., Han, I., and Ye, J. C. (2022). DiffuseMorph: Unsupervised Deformable Image Registration Using Diffusion Model. pages 347–364.

[68] Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*.

[69] Kingma, D. P. and Welling, M. (2014). Auto-Encoding Variational Bayes. *International Conference on Learning Representations (ICLR)*.

[70] Kipf, T. N. and Welling, M. (2019). Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pages 1–14.

[71] Le, B. H. and Deng, Z. (2012). Smooth Skinning Decomposition with Rigid Bones. *ACM Trans. Graph*, 31(6):1–10.

[72] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., and Shi Twitter, W. (2017). Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690.

[73] Lehoucq, R. B., Sorensen, D. C., and Yang, C. (1998). *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. Society for Industrial and Applied Mathematics.

[74] Lemeunier, C., Denis, F., Lavoué, L., and Dupont, F. (2023). SpecTrHuMS: Spectral transformer for human mesh sequence learning. *Computers & Graphics*, 115:191–203.

[75] Lescoat, T., Liu, H., Thiery, J., Jacobson, A., Boubekeur, T., and Ovsjanikov, M. (2020). Spectral Mesh Simplification. *Computer Graphics Forum*, 39(2):315–324.

[76] Lewis, J. P., Cordner, M., and Fong, N. (2000). Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, pages 165–172.

[77] Li, H., Weise, T., and Pauly, M. (2010). Example-based facial rigging. *ACM Transactions on Graphics*, 29(4):1–6.

[78] Li, R., Bladin, K., Zhao, Y., Chinara, C., Ingraham, O., Xiang, P., Ren, X., Prasad, P., Kishore, B., Xing, J., and Li, H. (2020). Learning Formation of Physically-Based Face Attributes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3410–3419.

[79] Li, T., Bolkart, T., Black, M. J., Li, H., and Romero, J. (2017). Learning a model of facial shape and expression from 4d scans. *ACM Trans. Graph.*, 36(6):194–1.

[80] Lin, J., Yuan, Y., Shao, T., and Zhou, K. (2020). Towards High-Fidelity 3D Face Reconstruction from In-the-Wild Images Using Graph Convolutional Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5891–5900.

[81] Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., and Black, M. J. (2015). SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16.

[82] Melzi, S., Ren, J., Rodolà, E., Sharma, A., Wonka, P., and Ovsjanikov, M. (2019). ZoomOut: spectral upsampling for efficient shape correspondence. *ACM Trans. Graph*, 38(6):155: 1–14.

[83] Melzi, S., Rodol, E., Castellani, U., and Bronstein, M. (2018). Localized Manifold Harmonics for Spectral Shape Analysis. *Computer Graphics Forum*, 37(6):20–34.

[84] Meyer, M., Desbrun, M., Schröder, P., and Barr, A. H. (2003). Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. *Visualization and Mathematcs III*, page 35–57.

[85] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2021). Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106.

[86] Moreau, A., Song, J., Dhamo, H., Shaw, R., Zhou, Y., and Pérez-Pellitero, E. (2024). Human gaussian splatting: Real-time rendering of animatable avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 788–798.

[87] Murray, J. D. and VanRyper, W. (1996). *Encyclopedia of Graphics File Formats*. O'Reilly & Associates, Inc., Sebastopol, second edition.

[88] Nasikun, A. and Hildebrandt, K. (2022). The Hierarchical Subspace Iteration Method for Laplace–Beltrami Eigenproblems. *ACM Trans. Graph.*, 41(2):17: 1–14.

[89] Neumann, T., Varanasi, K., Wenger, S., Wacker, M., Magnor, M., and Theobalt, C. (2013). Sparse localized deformation components. *ACM Transactions on Graphics*, 32(6):1–10.

[90] Nichol, A. and Dhariwal, P. (2021). Improved Denoising Diffusion Probabilistic Models.

[91] Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., and Chen, M. (2021). GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models.

[92] Ortega, A., Frossard, P., Kovacevic, J., Moura, J. M., and Vandergheynst, P. (2018). Graph Signal Processing: Overview, Challenges, and Applications. *Proceedings of the IEEE*, 106(5):808–828.

[93] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., and Antiga, L. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

[94] Pidhorskyi, S., Adjeroh, D. A., and Doretto, G. (2020). Adversarial Latent Autoencoders. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*.

[95] Popović, J., Seitz, S. M., and Erdmann, M. (2003). Motion sketching for control of rigid-body simulations. *ACM Transactions on Graphics*, 22(4):1034–1054.

[96] Qian, S., Kirschstein, T., Schoneveld, L., Davoli, D., Giebenhain, S., and Nießner, M. (2024). Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20299–20309.

[97] Qiao, Y., Gao, L., Yang, J., Rosin, P., Lai, Y., and Chen, X. (2022). Learning on 3D Meshes With Laplacian Encoding and Pooling. *IEEE Transactions on Visualization and Computer Graphics*, 28(2):1317–1327.

[98] Rabby, A. and Zhang, C. (2023). Beyondpixels: A comprehensive review of the evolution of neural radiance fields. *arXiv preprint arXiv:2306.03000*.

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

[100] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

[101] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. (2022). Hierarchical Text-Conditional Image Generation with CLIP Latents. (Figure 3).

[102] Ranjan, A., Bolkart, T., Sanyal, S., and Black, M. J. (2018). Generating 3D Faces Using Convolutional Mesh Autoencoders. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pages 725–741. ECCV.

[103] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-Resolution Image Synthesis with Latent Diffusion Models. pages 10674–10685.

[104] Russo, M. (2010). *Polygonal Modeling: Basic and Advanced Techniques*. Jones & Bartlett Learning.

[105] Saharia, C., Chan, W., Chang, H., Lee, C., Ho, J., Salimans, T., Fleet, D., and Norouzi, M. (2022a). *Palette: Image-to-Image Diffusion Models*, volume 1. Association for Computing Machinery.

[106] Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D. J., and Norouzi, M. (2022b). Image Super-Resolution Via Iterative Refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[107] Saito, S., Schwartz, G., Simon, T., Li, J., and Nam, G. (2024). Relightable gaussian codec avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141.

[108] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved Techniques for Training GANs. *Advances in neural information processing systems*, pages 2234–2242.

[109] Sanyal, S., Bolkart, T., Feng, H., and Black, M. J. (2019). Learning to Regress 3D Face Shape and Expression from an Image without 3D Supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7763–7772.

[110] Sekuler, R. and Blake, R. (2006). *Perception*. McGraw-Hill, Boston, 5th edition.

[111] Shamai, G., Slossberg, R., and Kimmel, R. (2019). Synthesizing Facial Photometries and Corresponding Ge-ometries Using Generative Adversarial Networks. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 15(3s):1–24.

[112] Sharp, N. and Crane, K. (2020). A Laplacian for Nonmanifold Triangle Meshes. *Computer Graphics Forum*, 39(5):69–80.

[113] Slossberg, R., Shamai, G., and Kimmel, R. (2018). High Quality Facial Surface and Texture Synthesis via Generative Adversarial Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[114] Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. *32nd International Conference on Machine Learning, ICML 2015*, 3:2246–2255.

[115] Sorkine, O. (2005). Laplacian Mesh Processing. *Eurographics (STARs)*, 29.

[116] Sorkine, O., Alexa, M., and Berlin, T. U. (2007). As-Rigid-As-Possible Surface Modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116. Alexander Belyaev.

[117] Sorkine, O. and Cohen-Or, D. (2004). Least-squares Meshes. In *Proceedings Shape Modeling Applications IEEE*, pages 191–199.

[118] Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., and Seidel, H.-P. (2004). Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184.

[119] Sorkine, O., Cohen-Or, D., and Toldeo, S. (2003). High-pass quantization for mesh encoding. *Eurographics Symposium on Geometry Processing*, 42:42–51.

[120] Sumner, R. W. and Popovic, J. (2004). Deformation Transfer for Triangle Meshes. *ACM Transactions on Graphics*, 23(2):399–405.

[121] Sumner, R. W., Zwicker, M., Gotsman, C., and Popovi´c, J. P. (2005). Mesh-Based Inverse Kinematics. *ACM transactions on graphics (TOG)*, 24(3):488–495.

[122] Tan, Q., Gao, L., Lai, Y. K., and Xia, S. (2018). Variational Autoencoders for Deforming 3D Mesh Models. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 5841–5850.

[123] Tena, J. R., De La Torre, F., and Matthews, I. (2011). Interactive Region-Based Linear 3D Face Models. *ACM SIGGRAPH 2011 papers*, pages 1–10.

[124] Thies, J., Zollhöfer, M., Nießner, M., Valgaerts, L., Stamminger, M., and Theobalt, C. (2015). Real-time Expression Transfer for Facial Reenactment. *ACM Trans. Graph.*, 34(6):183:1–183:14.

[125] Tong, W., Yang, X., Pan, M., and Chen, F. (2020). Spectral mesh segmentation via ℓ_0 gradient minimization. *IEEE Trans. Vis. Comput. Graph.*, 26(4):440–456.

[126] Tuan Tran, A., Hassner, T., Masi, I., Paz, E., Nirkin, Y., and Medioni, G. (2018). Extreme 3D Face Reconstruction: Seeing Through Occlusions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3935–3944.

[127] Váša, L. and Rus, J. (2012). Dihedral Angle Mesh Error: a fast perception correlated distortion measure for fixed connectivity triangle meshes. *Eurographics Symposium on Geometry Processing*, 31(5):1715–1724.

[128] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, , and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 2017-Decem(Nips):5999–6009.

[129] Verma, N., Boyer, E., and Verbeek, J. (2018). FeaStNet: Feature-Steered Graph Convolutions for 3D Shape Analysis. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2598–2606.

[130] Vlasic, D., Brand, M., Pfister, H., and Popovic, J. (2005). Face Transfer with Multilinear Models. *ACMTrans. Graph*, 24(3):426–433.

[131] Wang, H., Lu, T., Au, O., and Tai, C. (2014). Spectral 3D mesh segmentation with a novel single segmentation field. *Graphical Models*, 76(5):440–456.

[132] Wang, K., Torkhani, F., and Montanvert, A. (2012). A fast roughness-based approach to the assessment of 3D mesh visual quality. *Computers and Graphics (Pergamon)*, 36(7):808–818.

[133] Wang, X., Guo, Y., Deng, B., and Zhang, J. (2020). Lightweight Photometric Stereo for Facial Details Recovery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 740–749.

[134] Wu, C., Bradley, D., Gross, M., and Beeler, T. (2016). An Anatomically-Constrained Local Deformation Model for Monocular Face Capture. *ACM Trans. Graph.*, 35(12):1–115.

[135] Wu, F., Bao, L., Chen, Y., Ling, Y., Song, Y., Li, S., Ngi Ngan, K., and Liu, W. (2019). MVF-Net: Multi-View 3D Face Morphable Model Regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 959–968.

[136] Wu, Q., Zhang, J., Lai, Y. K., Zheng, J., and Cai, J. (2018). Alive Caricature from 2D to 3D. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 7336–7345. IEEE.

[137] Wu, S., Rupprecht, C., and Vedaldi, A. (2020a). Unsupervised Learning of Probably Symmetric Deformable 3D Objects from Images in the Wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1–10.

[138] Wu, Y. and Ji, Q. (2019). Facial Landmark Detection: a Literature Survey. *International Journal on Computer Vision*, 127(2):115–142.

[139] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. (2020b). A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21.

[140] Xiao, Y. P., Lai, Y. K., Zhang, F. L., Li, C., and Gao, L. (2020a). A survey on deep geometry learning: From a representation perspective. *Computational Visual Media*, 6(2):113–133.

[141] Xiao, Y. P., Lai, Y. K., Zhang, F. L., Li, C., and Gao, L. (2020b). A survey on deep geometry learning: From a representation perspective. *Computational Visual Media*, 6(2):113–133.

[142] Xu, C., Lin, H., Hu, H., and He, Y. (2021). Fast calculation of Laplace-Beltrami eigenproblems via subdivision linear subspace. *Computers & Graphics*, 97:236–247.

[143] Yadan, O. (2019). Hydra - A framework for elegantly configuring complex applications.

[144] Yang, H., Zhu, H., Wang, Y., Huang, M., Shen, Q., Yang, R., and Cao, X. (2020). FaceScape: a Large-scale High Quality 3D Face Dataset and Detailed Riggable 3D Face Prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 601–610.

[145] Yuan, Y.-J., Lai, Y.-K., Yang, J., Fu, H., and Gao, L. (2019). Mesh Variational Autoencoders with Edge Contraction Pooling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 274–275. IEEE.

[146] Zeng, X., Peng, X., and Qiao, Y. (2019). DF2Net: A dense-fine-finer network for detailed 3D face reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2315–2324. IEEE.

[147] Zhang, H., Kaick, O. V., and Dyer, R. (2010). Spectral mesh processing. *Computer graphics forum*, 29(6):1865–1894.

[148] Zhang, J., Cai, H., Guo, Y., and Peng, Z. (2020). Landmark Detection and 3D Face Reconstruction for Caricature using a Nonlinear Parametric Model. *CoRR*, abs/2004.0:1–12.

[149] Zhao, J., Mathieu, M., and Lecun, Y. (2017). Energy-based generative adversarial network. *ICLR*.

[150] Zhou, Y., Deng, J., Kotsia, I., and Zafeiriou, S. (2019). Dense 3D Face Decoding over 2500FPS: Joint Texture & Shape Convolutional Mesh Decoders. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1097–1106.

[151] Zhou, Y., Wu, C., Li, Z., Cao, C., Ye, Y., Saragih, J., Li, H., and Sheikh, Y. (2020). Fully Convolutional Mesh Autoencoder using Efficient Spatially Varying Kernels. *Advances in Neural Information Processing Systems*, 33:9251–9262.

[152] Zhou, Z., Ma, F., Fan, H., Yang, Z., and Yang, Y. (2024). Headstudio: Text to animatable head avatars with 3d gaussian splatting. In *European Conference on Computer Vision*, pages 145–163. Springer.

[153] Zollhöfer, M., Thies, J., Garrido, P., Bradley, D., Beeler, T., Pérez, P., Stamminger, M., Nießner, M., and Theobalt, C. (2018). State of the art on monocular 3D face reconstruction, tracking, and applications. *Computer Graphics Forum*, 37(2):523–550.

This page is intentionally left blank.

APPENDIX A

DEEP3DMM COMPARISON PLATFORM SOFTWARE DESIGN

A.1 Overview

This appendix provides insight into software design and implementation details of the comparison platform.

A.1.1 Models

The methods are implemented as either autoencoder [129, 14, 46, 43] or variational autoencoder [151]. These models correspond to GraphVariationalAutoencoder and GraphAutoencoder classes in Figures A.1 and A.2. These classes inherit from the LightningModule [38], significantly reducing the implementation boilerplate.

A.1.2 Architectures

The models are composed of different pieces of neural network architectures. As shown in Figure A.1, the GraphAutoencoder comprises one instance of the MeshEncoder class and one instance of the MeshDecoder class. These are subclasses of the torch.nn.Module class, which is a base class for all neural network modules in PyTorch framework [93]. The GraphVariationalAutoencoder has a different encoder, an instance of the ProbabilisticMeshEncoder class. Unlike the MeshDecoder, which outputs the latent code $\mathbf{Z}$, the ProbabilisticMeshEncoder outputs the mean and standard deviation of the normal distribution, from which GraphVariationalAutoencoder samples the latent code input to an instance of the MeshDecoder.

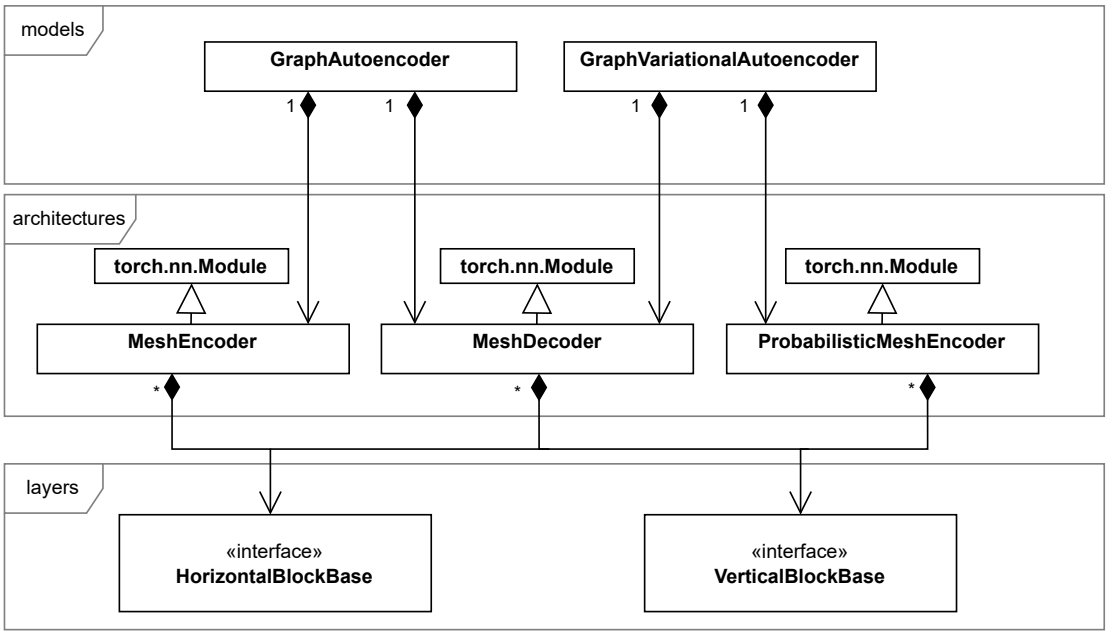


Figure A.1 The UML diagram showing the relationship between classes contained within three core modules of the Deep3DMM Comparison Platform: layers, architectures and models. Sequences of layers form HorizontalBlock or VerticalBlock objects composed into architectures such as MeshEncoder, MeshDecoder and ProbabilisticMeshDecoder. These architectures build the model, either an autoencoder or a variational autoencoder. This diagram considerably simplifies the layers module, and Figure A.3 gives more insight into it. The models module is also simplified in this diagram. Figure A.2 provides more details on this module.

A.1.3 Horizontal and vertical blocks

The architectures are composed of instances of classes which implement the HorizontalBlockBase and the VerticalBlockBase abstract classes. Although these classes can be confusing at first, they improve the computational and memory footprint of the implementation. In some cases, they allow to reduce the number of learnable parameters.

Horizontal blocks work with graphs that have the same same number of vertices and connections, and contain operations which do not change the graph structure. Because of this, they only need one graph sampler, and learned mappings between graphs can be reused, making the model more efficient. Vertical blocks, on the other hand, handle changes in graph structure, such as upsampling, downsampling, and convolutions with strides greater than 1, meaning that they require separate graph samplers for each operation. This approach improves flexibility by performing graph sampling during runtime instead of requiring predefined structures in preprocessing.

Figure A.3 depicts intricacies of HorizontalBlockBase and VerticalBlockBase classes.

A.1.4 Layers and samplers

The horizontal and vertical blocks are built of neural network layers. The type of these layers is the main differentiator between the compared methods. For example, each method uses a different convolutional operator. Additionally, the horizontal and vertical blocks contain an instance or multiple instances of classes implementing a GraphSamplerBase abstract class. These instances perform the graph downsampling or upsampling operation and store the attributes associated with graph sampling.

A.1.5 Data processing and training

Figure A.2 shows the main components involved in the training process of deep 3D morphable models. Apart from the previously described pieces of neural architecture, the models use a GeometricLosses object, which provides methods for calculating the error metrics used in a loss function. The model uses the UniversalDatastructure as an intermediary to access data stored on a hard drive. The ExplicitDataModule uses the same UniversalDatastructure to get paths to data, including samples from the training dataset.

The model, such as a GraphAutoencoder or a GraphVariationalAutoencoder, the ExplicitDatamodule and different types of callbacks and loggers are used by the pytorch_lightning.Trainer [38], which runs the training, validation and test dataloaders, handles the forward pass and the backpropagation, ensures that data is on the correct device and calls the callbacks. The logger tracks the metrics and other information through

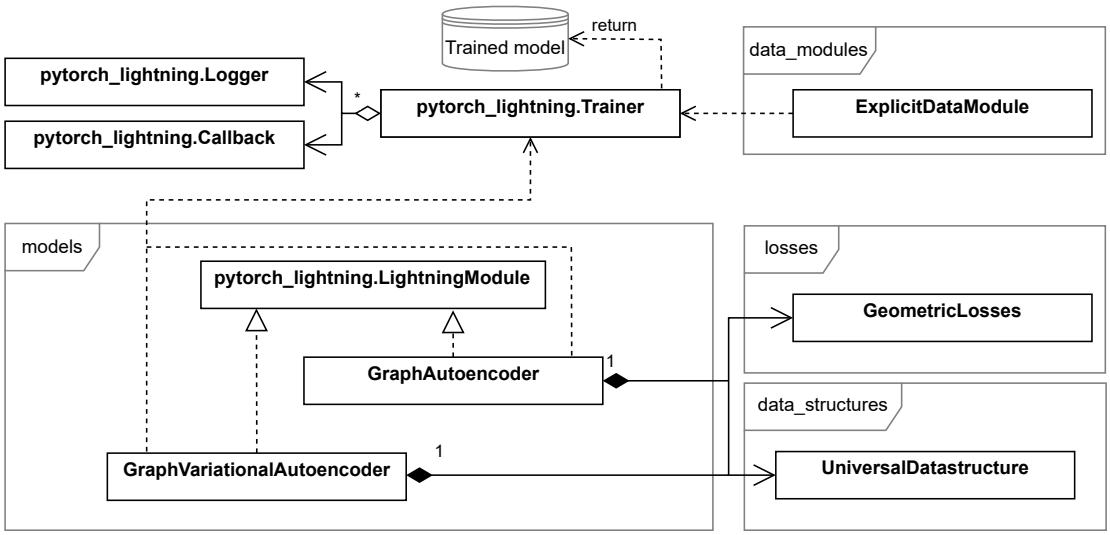


Figure A.2 The UML diagram depicting the relationship between classes of the Deep3DMM Comparison Platform, which take part in training, testing and validation of the deep 3D morphable model. Each model contains the GeometricLosses object, which provides methods for the error metrics calculation. Each model also has an instance of the UniversalDatastructure, allowing the model to access external data and metadata independently of the file structure of the dataset. The pytorch_lightning.Trainer object orchestrates the training. It runs the forward process and the backpropagation of the model, calls the callbacks and logs the results using a choice of loggers. It uses an ExplicitDataModule to lazy-load batches of training, validation and test data into the model. The optimised parameters, the associated hyperparameters and other configurations are stored on the hard drive. The details on the objects of which the models are composed are shown in Figure A.3. The relationships with other modules of the data_structures and data_modules can be found in Figure A.5.

the appropriate callbacks. In this work, the Weights and Biases [10] experiment tracking platform is used to log and visualise the metrics gathered during the training process.

A.2 Layers and architectures

A.2.1 Graph sampling

The GraphSamplerBase abstract class is an interface for different graph sampling methods. Quadric Mesh Simplification [44] is performed by methods in the QuadricMeshSimplification class. This sampling method is based on topology and coordinates of sampled meshes. In contrast, simple sampling proposed in [151] is based purely on graph connectivity, and it results in a sequence of non-manifold graphs. Simple sampling is performed by methods in the SimpleSampler class.

A.2.2 Graph convolution and feature aggregation

For the purpose of convolution operation, the lower resolution graph is calculated using the SimpleSampler class. Consistently with traditional convolutional operators, transpose graph convolution allows to upscale a graph. SpiralConv [14], SpiralPlusPlusConv [46], FeaStConv [129], VCConv [151] and LSAConv [43] are subclasses of torch.nn.Module. Each of these classes implements its corresponding graph convolution method.

Features aggregation is implemented in SpiralConv [14], SpiralPlusPlusConv [46], FeaStConv [129], VCConv [151] and LSAConv [43] classes. The residual block is implemented as part of the VCConv [151] class.

A.2.3 Horizontal and vertical blocks

In object-oriented programming, the object can be interacted with invocation and by inspection. Invocation refers to interacting with an object by calling its methods. That includes polymorphism, where the same method invoked by a different object type can run a different code. Inspection refers to externally examining and using the object based on this information. In object-oriented programming, invocation is preferred over inspection because there are often no formal ways to test if the object satisfies the user's requirements. The abstract base classes (ABC) were introduced to overcome this issue in Python programming language. Abstract base classes allow the definition of a consistent interface for every class which inherits from them. Consequently, they ensure that classes, which inherit from the abstract base class, implement all the required abstract methods and properties.

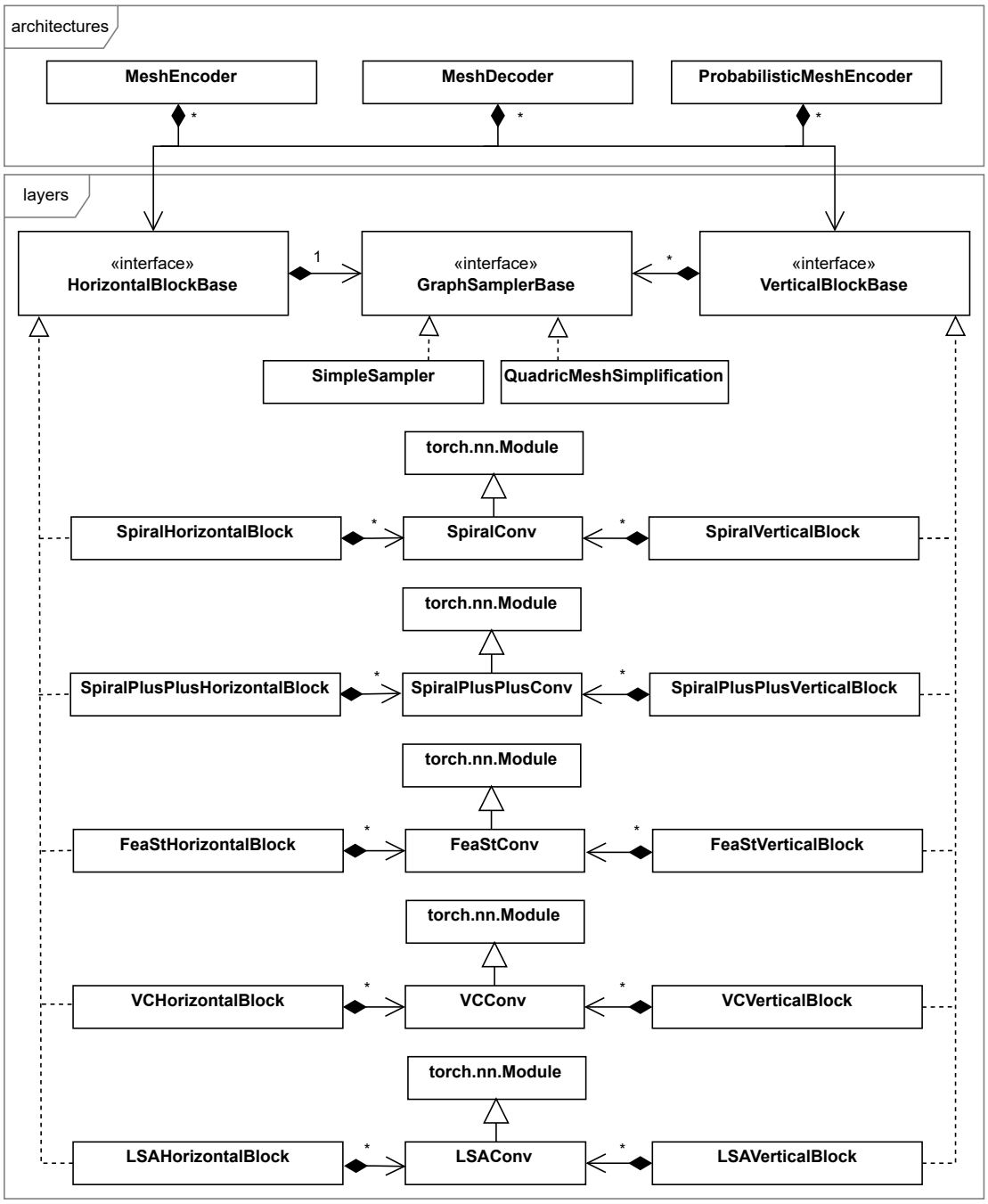


Figure A.3 The UML diagram showing the relationship between the classes in the layers module of the Deep3DMM Comparison Platform. SpiralConv [14], SpiralPlusPlusConv [46], FeaStConv [129] and LSAConv [22] implement the graph convolutions and pooling layers, while VCConv [151] implements the residual blocks built of graph convolutions, pooling and residual layers. The horizontal and vertical blocks are the compositions of these objects. The HorizontalBlocks have only one graph sampler instance, while VerticalBlocks have at least one graph sampler.

MeshEncoder, MeshDecoder and ProbabilisticMeshEncoder classes contain a sequence of horizontal blocks or vertical blocks. To ensure that all the required methods are implemented and that they behave predictably, different types of horizontal and vertical blocks are implementations of base abstract classes, the HorizontalBlockBase and the VerticalBlockBase. Each horizontal or vertical block contains at least one Conv class associated with the type of this horizontal or vertical block. For example, SpiralHorizontalBlock contains at least one instance of SpiralConv [14] class.

The original implementations of the methods compared in this work rely on the separate preprocessing step, in which graph sampling is performed. Performing this graph sampling preprocessing step requires prior knowledge about the number of subsampled and upsampled graphs. In contrast, graph sampling is performed during runtime in the proposed implementation based on the model's structure. Therefore, the proposed design provides more flexibility in experimentation.

Importantly, subclasses of the HorizontalBlockBase have only one instance of a class implementing the GraphSamplerBase abstract class because horizontal blocks operate on a sequence of graphs with the same connectivity and vertex count. In other words, the horizontal blocks cannot contain any layers that change the graph structure: upsampling layers, downsampling layers or convolutional and transposed convolutional operations with strides greater than 1. This design decision allows the use of only one graph sampler for each operation within the horizontal block. Additionally, in methods which use learned mapping between the consecutive graphs [151, 43], this learned mapping can be reused within the horizontal block because the structure of the graphs is the same. This design comes with two advantages. Firstly, the number of learnable parameters can be reduced when using methods which utilise the learned mapping between the consecutive graphs. Secondly, regardless of the method used, the mapping between the consecutive layers can be calculated once and shared between the layers, reducing memory consumption.

Unlike the horizontal blocks, the vertical blocks handle graph upsampling and downsampling, as well as the convolutional and transposed convolutional operations with strides of 2 and more. Each of these operations is associated with a separate instance of a class which implements the GraphSamplerBase abstract class.

A.3 Data loading and processing

A.3.1 Universal interface to different file structures

Datasets of facial meshes can be stored in various incompatible directory structures and naming conventions, as they often come from external sources. The DatastructureBase abstract class addresses this incompatibility issue. The class provides a unified interface to

datasets and to other processing and neural network training metadata. The methods of the DatastructureBase abstract class allow iteration over dataset samples by subject or by shape, independently of the actual structure of the data, as long as there exists a class which implements the DatastructureBase abstract class. In this work, the UniversalDatastructure class implements the DatastructureBase. This UniversalDatastructure class is compatible with the file structure presented in Figure A.4.

UniversalDatastructure is used whenever there is a need to access or save any data or metadata. It is used by the model, the data modules which load the training, validation and test batches, as well as by the implementations of the DataProcessBase abstract class, which preprocess the datasets. The file structure shown in Figure A.4 is the result of multiple iterations and has proven sufficiently flexible and practical in this work.

The data directory contains subdirectories for each dataset. Each dataset directory organises files into three main subdirectories: _general, _training and _representations. The _general directory stores information common to all the other files in the dataset, regardless of the data representation. The textfiles containing the names of shapes and subjects are used by the ExplicitDataModule class and different iterators, which are depicted in Figure A.5. The tri_faces.npy and quad_faces.npy files store the connectivity information of dataset samples in NumPy format. As the mesh connectivity is shared across all the dataset samples, storing this information in each sample would be redundant and inefficient, considering both memory footprint and data loading time during the model training.

The _representations directory stores different representations of the dataset samples in separate subdirectories. These representations are the signal on the graph's nodes and are used as inputs to deep 3D morphable models. They also play a role in the calculation of the loss functions, as shown in Equations (3.20 - 3.25). Each of them is divided into directories for different facial shapes, and each file within each shape directory contains NumPy array files, one per subject. Additional information, such as a mean face, standard deviation or minimum and maximum features, is stored as NumPy array files with an underscore as a prefix. Different representations are the results of different stages of data preprocessing, described with Equations (3.14 - 3.19).

The _training directory contains the training versions subdirectories. Each training version stores the text files used for model training. In the work covered in this chapter, the input_shapes.txt and output_shapes.txt contain the same neutral shape name. The latent_points.txt specifies the nodes of the bottleneck layer in Mesh Autoencoder [151] method. Finally, the training_subjects.txt, validation_subject.txt and testing_subjects.txt are output by a method of the DataProcess class. The method splits the subjects listed in all_subject_names.txt into training, validation and test sets. The same set of training, validation and test samples across all the experiments ensures a fair comparison.

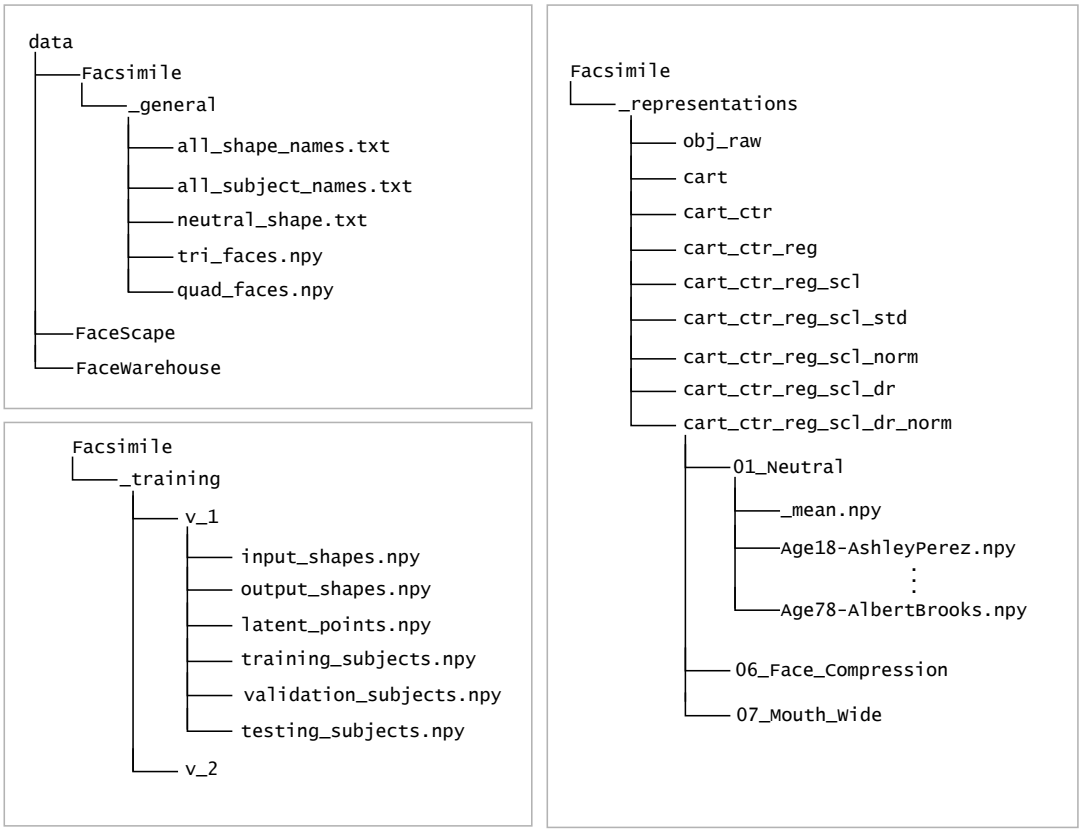


Figure A.4 The structure of datasets and other files used in this work by the Deep3DMM Comparison Platform. The UniversalDatastructure provides an interface between this file structure and the objects of the platform.

Convenient orchestration of data processing

The DataProcessBase abstract class implements methods for common data processing operations. These methods use iterators with associated readers and transforms. The resulting interface allows quickly composing an arbitrary data processing pipeline. The methods must be listed in the `process()` method of the DataProcess subclass. This method calls a sequence of the processing operations defined by the parent class.

A.3.2 Iterators

The data processing pipeline is built of a sequence of iterators. The iterators inherit from the torch.utils.data.Dataloader [93] class, and they accommodate iteration through the set of data samples stored in either OBJ or NumPy [51] format. Different types of iterators are implemented, depending on the metric calculated across all the iterated samples. The Numpy iterator object is the basic iterator, which does not calculate or output any metrics. The iterator instances of the Mean, Scale, MinMax, MinMaxMean, StdDev and L1Norm classes output the following, respectively: per-channel average of the samples, uniform scalar factor calculated as in Equation (3.7), the minimum and the maximum per-channel values of a feature, minimum, maximum and average per-channel values of a feature and the L_1 norm calculated between two samples. The metrics, calculated and saved after the iteration is complete, can be used by the transformation objects in the subsequent iterator.

A.3.3 Readers and data formats

The raw dataset samples are stored in Wavefront OBJ format [87]. This format encodes the mesh geometry as ASCII text; therefore, its read and write time is longer than that of binary formats. Moreover, the Wavefront OBJ files store the faces information in each sample, despite being the same across all the samples in the dataset. Based on the comparative evaluation, read and write time of the same meshes stored in the OBJ, STL and NumPy formats, NumPy format significantly outperforms the other formats. Read and write time is especially important in this work because the training samples are lazy-loaded by the instance of the ShapesReader class, which is part of the ExplicitDataModule. As all the training data cannot fit in a graphics processing unit (GPU), lazy-loading allows one to read and store only a batch of data samples at one time in memory. This requires repeated access to training data files, and consequently, the read time of the files can accumulate significantly. The meshes are converted to NumPy arrays before any further preprocessing steps to decrease the data processing time.

The readers inherit from the torch.utils.data.Dataset class. The readers inherit from the torch.utils.data.Dataset class. The iterators use them to accommodate the reading

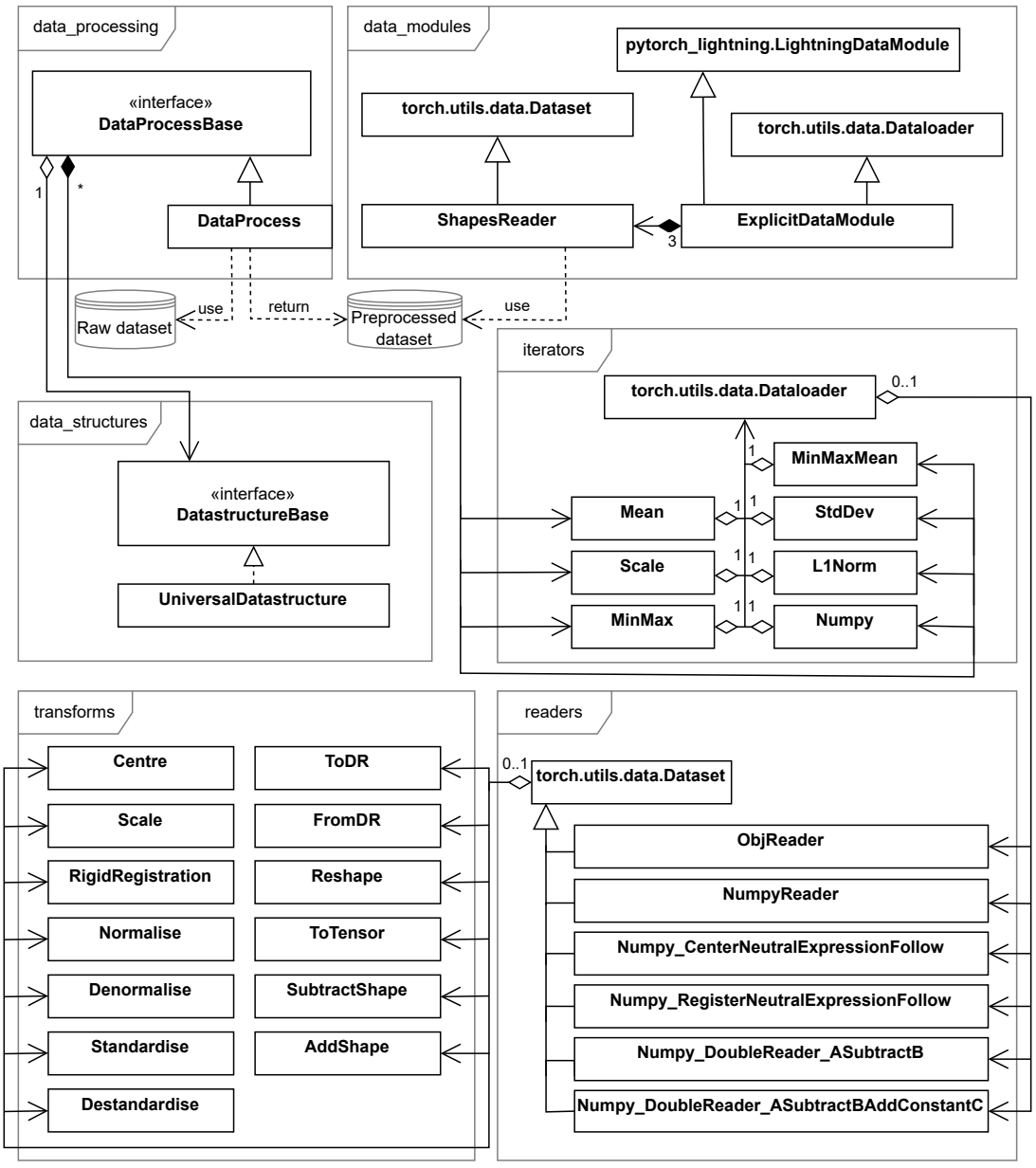


Figure A.5 The UML diagram showing the relationship between the classes of the modules responsible for data loading and data processing. The data processing pipeline is built of a sequence of iterators, which can calculate different metrics over the set of iterated samples. The iterators use the readers to accommodate the reading of various file formats. A sequence of transforms can be applied to each sample by the reader. The DataProcessBase abstract class implements methods for common data processing operations. These methods use iterators with associated readers and transforms. The DataProcess subclass implements the process() method, which calls a sequence of the processing operations defined by the parent class. The resulting preprocessed dataset is used by the ExplicitDataModule in the training, testing and validation process, as shown in the diagram in Figure A.2.

of different file formats. Wavefront OBJ and NumPy files are supported by ObjReader and NumpyReader. Apart from loading the content of a file, the reader can apply a transformation to each of the loaded samples. The transforms are used by the readers and applied to each sample. Those implemented as objects have an advantage over functions. Namely, at the initialisation stage, they allow for pre-computation of the attributes used in the transformation. Furthermore, the metrics output by the iterator from the previous preprocessing iteration can be loaded and processed at the initialisation.

It is preferred to iterate through the dataset as few times as possible. Consequently, given the number of data transformations to be performed, it is preferred to stack as many of them as possible in one reader. Unfortunately, the minimum number of iterations over the whole dataset is determined by the number of metrics that need to be calculated over the whole dataset. For example, standardisation requires the mean and standard deviation output from the previous iteration. Thus, it cannot be performed after another transformation in the reader.

This page is intentionally left blank.

GLOSSARY

AAA Pronounced triple-A. A lot of time, A lot of resources, A lot of money. In the video games industry, the term used to classify a high-budget, popular game from a major publisher. 2

Action Unit Standardised unit of facial expression defined within the Facial Action Coding System (FACS) that represents a specific facial muscle movement or combination of movements. 165, 167

canthus Corner of the eye where the upper and lower eyelids meet. 84

deep 3D Morphable Model Computational framework that employs deep learning techniques to represent and manipulate 3-dimensional facial shapes, allowing for the generation, editing, and analysis of facial geometry. 167

EKER™ Software developed by Humain Ltd. EKER™ is a facial rigging system based on Facial Action Coding System. 2

Facial Action Coding System Comprehensive framework used to objectively categorise and describe human facial expressions based on the underlying muscular movements. 2, 165, 167

Humain Ltd. Industrial partner of this research. Headquartered in Belfast, the company provides rigging services to world leading entertainment and technology companies within the video and games industry. 2, 165

jowl The loose skin and flesh under the jaw, which most people develop with age. 84

neuron's receptive field In physiology, the region of the retina over which visual signals influence the activity of that neuron. 71

neutral face Baseline position of a person's face, denoted as AU 0. Facial pose which shows no evidence of an Action Unit. It exhibits usual shapes of the features and permanent wrinkles. 2

This page is intentionally left blank.

ACRONYMS

AU Action Unit 165, *Glossary:* Action Unit

Deep 3DMM Deep 3D Morphable Model *Glossary:* deep 3D Morphable Model

FACS Facial Action Coding System *Glossary:* Facial Action Coding System