Imperial College London
Department of Computing

Advances of Graph Neural Networks for 3D shape learning and analysis

Rolandos Alexandros Potamias

September 2023

Supervised by Dr. Stefanos Zafeiriou

Submitted in part fulfilment of the requirements for the degree of PhD in Computing and the Diploma of Imperial College London. This thesis is entirely my own work, and, except where otherwise indicated, describes my own research.

Abstract

In recent years, there has been a surge in the availability of 3D sensors, leading to an exponential increase in the amount of 3D data, paving the way for remarkable advancements in 3D computer vision applications. The advances of deep learning on geometric data with irregular structures, such as meshes and point clouds, have further enhanced the ability to analyze and understand 3D shapes.

In this thesis, we explore the usage of geometric deep learning methods in 3D shape analysis. The thesis can be divided into two parts. In the first part, the advancements of non-linear localized mesh convolutions to tackle the limitations of traditional statistical shape modeling methods, such as PCA, to capture fine details and extreme deformations given its linear structure and global structure is shown. In particular, the cases of static and dynamic neural deformable models are explored using a localized mesh convolution operator to generate high fidelity deformable models. Using a large scale dataset of hand scans composed by over 1200 subjects, a fine-grained neural hand model is constructed that is able to outperform current state-of-the-art hand models. To explore the expressive power of graph convolution in dynamic morphable models, a 4D generative model is proposed that is able to manipulate 3D faces and generate dynamic expressions fully customized by the user. Both models achieve state-of-the-art performance that outperforms traditional PCA deformable models.

However, dealing with high-fidelity models poses several challenges, especially when it comes to processing and storage. The enormous amount of points needed to capture the fine-grained features of such models can be computationally expensive and memory-intensive which limits their real-time applications. In the second part of this thesis, two neural based simplification methods are proposed to simplify point clouds and meshes in real-time. Both methods rely on graph neural networks to capture rich local and global topological information of the 3D objects. Initially, a point cloud simplification method is proposed that samples points in an sophisticated matter to preserve the underlying perceptual features of the point cloud. Then, the simplification method is extended to meshes through a graph neural network triangulation module that constructs the faces of the simplified mesh. Through extensive evaluations and comparisons with state-of-the-art baselines, we demonstrate the effectiveness and efficiency of our method in preserving important shape characteristics while significantly reducing the data size.

Acknowledgements

This work would not have been possible without the support of so many mentors, colleagues, supporters, family members, and friends over the years. First and foremost, I want to thank my supervisor Dr. Stefanos Zafeiriou for granting me the invaluable opportunity to embark on a PhD journey under his guidance and for providing me with an environment where my creativity could flourish. Stefanos provided me with the resources and the freedom to explore my research interests, allowing me to pursue my academic endeavors without hesitation.

My accomplishments wouldn't have been possible without the extensive discussions with my fellow colleagues. Over the years I had the chance to interact and collaborate with some amazing people that made my PhD experience a beatuful journey. I would like to thank my good friend and "partner in crime" Stylianos Ploumpis for the continuous collaboration we had throughout those years and for his thoughtful advises on several research problems. I would also like to thank my friends and collaborators throughout my PhD years, Evangelos Ververas, Stylianos Moschoglou, Georgios Bouritsas and Michael Tarasiou for the endless discussions and debates we had that enabled me to advance my thinking and develop my research quality. Of course, I cannot forget Jiali Zheng, that remained from my first day at Imperial, not only an invaluable collaborator but also a cherished friend over the years. I would like to thank her for her unwavering support and for the countless moments of laughter throughout the years. Additionally, I would like also to thank my colleagues at Imperial: Alexandros, Stathis, Margarita, Dinos, Phoivos, Dimitris, Michalis and Jiankang for making this journey all the more rewarding and enjoyable. I am grateful for the fun-filled memories we created together in the office.

A special thank you is also dedicated to my great friends outside the lab who provided steadfast support and understanding during the emotional roller coaster of my PhD journey. I am grateful for their presence, encouragement, and belief in me.

Lastly, my deepest gratitude goes to the most significant individuals in my life, my family. I am deeply thankful to my father, Giorgos, and my mother, Katerina, for their guidance and invaluable advice during the most challenging moments of my journey.

This thesis is dedicated to my parents,
Giorgos and ***Katerina***.

Contents

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

Contents

1.1 Problem Scope and Challenges

In the recent days, the growth of the internet, social media, and digital devices has led to an explosion of visual data, with millions of images and videos being uploaded and shared every day. The vast amount of available data along with the development of specialized hardware, such as graphics processing units (GPUs), has contributed to the rise of deep learning era. Deep learning has revolutionized several fields ranging from in natural language processing and machine translation [11, 12] to speech recognition and computer vision, enabling tremendous breakthroughs in image recognition [13], object detection [14, 15], and semantic segmentation [16]. Deep learning models are function estimation methods designed to learn hierarchical representations of visual data, allowing them to identify complex patterns and features in images, videos, and other visual data with unprecedented accuracy and efficiency. Usually they are composed by many layers that have the capacity to interpolate large-scale datasets.

However, the majority of deep learning methods that have been proposed apply to data with Euclidean structures, such as 1D sequences or 2D grids and do not translate to *ir-*

regular structured data. Initial approaches attempted to tackle irregular data, such as 3D shape, using euclidean representations in order to leverage well studied approaches such as Convolutional Neural Networks (CNNs). One of the most famous representations to approximate the 3D surface are 3D Voxel grids, that can be processed using volumetric CNNs, defined directly on the Euclidean grid [17, 18, 19]. However, the major disadvantages of volumetric approaches include their substantial computational complexity and imprecise representation, which can distort the topological characteristics of the shapes. Additionally, these methods exhibit a significant redundancy, as they explicitly model the interior of the shape, whereas in many applications, the focus is primarily on the surface. Such limitations enforced the generalization of deep learning on irregular structured data, such as graphs and manifolds. Nevertheless, their inherit lack the notion of ordering and distancing has a great impact on directly applying deep learning techniques to such structures. During the last years several methods have arisen that explore deep learning models on irregular domains that either completely lack structure, such as point sets, or lie on non-Euclidean domain, such as graphs and manifolds. The manifestation of deep learning models on irregular data has been established with the umbrella term "*Geometric Deep Learning*" [4]. Social networks that define the relations between social media users, the protein structure in a molecule as well as Riemannian manifolds that define 3D objects are only some of the examples that can not be modeled directly with traditional deep learning techniques. In this thesis we will only refer to three types of irregular data:

- **Point Clouds**: They belong in the category of sets, which are a mathematical model for an unorganized collection of items. In their simplest form they contain an unstructured set of points embedded on the 3D space. Point clouds are widely used given that they are lightweight and the most accessible representation of 3D surfaces. However, unlike grid-like data structures, point clouds lack any inherent ordering, making it difficult to apply traditional order-dependent machine learning techniques directly to them.

- **Manifolds and Meshes**: In mathematics, a *manifold* can be considered as a a topological space that generalizes a surface in a way that it is locally homeomorphic to a Euclidean space. In other words, even though manifolds may have a more complex global structure they retain the properties of Euclidean spaces in a local level for small neighborhoods around each point. Manifolds can have various dimensions and shapes, and they are often used to model real-world phenomena with complex structures, such as surfaces, spaces of shapes, or even the space of all possible solutions to a system of equations. *Meshes* are considered as 2D manifolds or discrete surfaces and they are constructed by a collection of points and faces that define the surface

approximation. They are usually triangular or quadrilaterals according to the formulation used to represent their faces. They are among the most popular ways to represent a 3D object in computer vision and graphics along with Point Clouds and Voxel grids. However, similar to Point Clouds, the lack of intrinsic ordering makes applying deep leaning non-trivial task. A comparison between Point Clouds, Meshes and Voxels is illustrated in Figure 1.1.

- **Graphs:** They can be considered as a broad representation category of such irregularly structured data. A graph, in its general form, describes a system of relations along with their corresponding interactions. The objects of the system are usually called nodes or vertices and their relations links or edges. Graphs can be directed or undirected, depending on whether its edges have a specific direction. In a directed graph, the edges have a direction that information flows, while in an undirected graph, the edges do not have a direction and messages from nodes are transmitted in both ways. They can also be weighted or unweighted, depending on whether the edges have a weight or value associated with them. In a weighted graph, the edges have a numerical weight, while in an unweighted graph, all edges have the same weight. Given that a large amount of real world problems that naturally take the form of graphs, they are widely used in computer science, including in areas such as network analysis, social network analysis, and optimization. They are also used in many real-world applications, such as transportation networks, electrical circuits, and molecular structures.

Recently, **Implicit Functions** have also received a lot of attention as a way to represent 3D objects. Implicit functions enable the encoding of complex 3D shapes and structures without the need for explicit surface representations or voxel grids. Instead, they define surfaces as the zero-level set of a continuous function, making it easier to generate, manipulate, and render intricate 3D objects. They offer a versatile and compact way to capture and work with 3D data, ultimately pushing the boundaries of what's possible in 3D modeling and visualization. However, similar to voxel grids, the computational complexity they introduce is analogous to the sampling rate which limits their application in large scale datasets. Additionally, given their ability to can represent arbitrary mesh topologies without leveraging the intrinsic shape correspondences, they tend to perform worse compared to mesh representations in fixed topology settings. Finally, one of the significant limitations of implicit functions is the need for a meshing step, often involving algorithms like Marching Cubes, to convert the implicit representation into a mesh format that can be rendered and processed by common graphics engines and software.

On the other side, 3D point clouds or point sets, have been treated using a shared func-

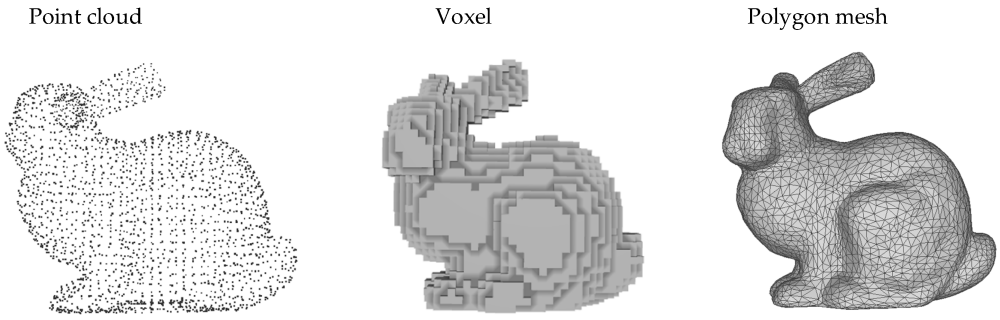


Figure 1.1: Various representations of a 3D bunny object. (Figure from [3])

tion over the points, which by construction hold the permutation equivariant property. In particular, both PointNet [20] and DeepSet [21] architectures have revolutionized the processes of point sets, achieving impressive performance on several benchmarks. Their basic idea is to learn a spatial encoding of each point and then aggregate all individual point features to a global point cloud signature. However, such formulation can only generate global representations and lack the ability to describe local structures and patterns. In an follow up work, PointNet++ [22], re-formulated the point cloud as a graph structure with edges locally connecting neighboring points. Such restructured approach could better imitate the concept of locality, that contributed to the success of Convolutional Neural Networks (CNNs) in Euclidean domain. PointNet++ achieved remarkable results and pioneered the formulation of Point Clouds as graphs. In the case of 3D meshes, the reformulation to a graph structure comes naturally given that each vertex can be considered as a node of a graph connected with a set of edges that form its faces. GeodesicCNN [5] initially generalized the convolutional networks (CNN) paradigm to non-Euclidean manifolds by constructing a local geodesic system of polar coordinates to extract "patches". The designed geodesic kernel was by construction invariant to deformations which was an intrinsic analogy of regular convolution. However, the fixed size and the ordering of each neighborhood, factors that enabled the weight sharing and the intrinsic correspondences between patches in CNNs, can not be trivially defined in non-Euclidean meshes. To address this, several methods have proposed to use fixed trajectories to count vertices around each neighborhood and then apply regular convolution [6, 7].

1.2 Objectives

The aim of this PhD is to explore the use and application of geometric deep learning and mesh convolutions to mitigate several issues of traditional methods of 3D shape analysis

and tackle common problems of mesh processing. This thesis can be divided in two parts. In the first part, two novel methods for 3D static and dynamic shape modeling are presented (Objectives 1 & 2) that aim to tackle the limitations of traditional shape modeling to capture high frequency non-linear details of a 3D surface using mesh convolutions. In the second part of the thesis, the storage processing requirements of such high fidelity 3D models are stressed and two novel graph neural network based methods for 3D shape simplification are proposed (Objectives 2 & 3). We evaluate their contribution to 3D meshes and point clouds under several applications, in terms of performance, compactness and runtime. In particular, the main objectives of this thesis are:

- **Objective 1**: To address the limitation of traditional 3D modeling to generate sharp shapes from compact representations. In particular, traditionally shape modeling, also called morphable modeling, is performed using Principal Component Analysis, which can only generate smooth meshes that lack details. In this thesis, we attempt to tackle this limitation of PCA 3D models by exploiting neural networks with mesh convolutions to create expressive 3D models.

- **Objective 2**: Similar to static 3D morphable models, the design of expressive dynamic morphable models that can generate realistic animations is also essential. A major limitation of statistical blendshape PCA models is their inability to generate extreme deformations. Till now, intense expressions and facial animation can only be generated by modifying artist-defined rigging. This makes modeling of human faces with extreme facial expressions especially challenging. In this thesis, we explore the power of mesh convolutions to create a framework that is able to generate photorealistic animation beyond traditional PCA blendshapes.

- **Objective 3**: Another challenge in the field of 3D computer vision is salience detection and simplification in 3D meshes and point clouds. Usually, 3D objects contain an enormous amount of points in order to be rendered with high detail. However, such information is usually redundant and affects the storage and processing requirements. In the literature, sampling and simplifying point clouds in a fast and efficient manner remains challenging. By utilizing graph neural networks we aim to explore both local and global patterns and overcome the limitations of the literature.

- **Objective 4**: Finally, another direction covered in this PhD is the implementation of a neural, graph-based, method that learns to triangulate 3D point clouds. Point cloud triangulation remains an unsolved problem that have been extensively studied in 3D community. Current methods either require an excessive amount of points or hand-crafted engineering to generate smooth results. Using the locality of graph

neural networks we aim to develop a method that can generalize to arbitrary shapes and accurately triangulate point clouds in real-time.

1.3 Contributions

In this section, we summarize the contribution of this thesis according to the aforementioned scope and objectives. The main objective of the work presented in this thesis is to utilize the advents of geometric deep learning and specifically mesh convolutions and graph neural networks in 3D shape analysis in order to overcome the limitations of traditional 3D modeling and simplification techniques. In particular

- In Chapter 3 we present the first large scale 3D hand model composed by over than 1200 subjects, which constitutes the first hand model trained with such large and diverse dataset. Currently, most methods that reconstruct and estimate human hand poses rely on the low polygon MANO model. However, this model has limited capability to capture diverse shape characteristics of real human hands because it was trained on only 31 adult subjects. Additionally, hand textures have been largely neglected in current methods. In this Chapter we address these issues by proposing a new shape and appearance hand model called "Handy++" trained on a large scale dataset with diverse ages, genders, and ethnicities. To train the shape component of Handy++ we utilize spiral convolutions which prove beneficial to model the details of the hand compared to traditional PCA models. Handy offers improved robustness and accuracy over existing methods. The contribution of the aforementioned shape method is not limited to hand shape modeling given that it can be easily applied to different types of 3D modeling such as faces and bodies and model sharper shapes compared to PCA models. Finally, we contribute to the community by making the model publicly available.

- In Chapter 4 we tackle the unexplored problem of dynamic generation of 3D meshes using mesh convolutions. In particular, although static 3D facial expression models have been widely studied, the generation of photorealistic facial animation remains relatively unexplored. In this Chapter, we implement a model that animates a static 3D face in a fully customized way. Using a large scale dataset composed of 180 subjects posing a series of different expressions, enables the generalization of the proposed method to any subject simply by switching the identity template. The contributions of this Chapter can be summarized in a) the introduction of a challenging task of dynamic facial expression generation, b) the creation of a versatile

generative model that can generate "a-la-cart" 4D expressions given a single mesh template and c) the implementation of a robust GNN decoder that generates sharp and extreme animations compared to blendshape models. Finally, we demonstrate that the proposed method can be utilized to animate and manipulate the expressions of any given subject from a single in-the-wild image.

- In Chapter 5 we introduce a learnable point cloud simplification method that learns to sample points from a point cloud according to their saliency. Traditional mesh and point cloud simplifications methods rely on time-consuming optimization schemes that iteratively sample or discard points according to their importance. However, in a real world scenario with hundreds of thousands points such scheme will set a large computational burden. In this Chapter, we rely on a series of studies that suggest that curvature related features highly correlate with the human perception and provide the salient cues of the 3D object. We construct a graph neural network based architecture that assigns an inclusion probability to each point of the input. The core of this method is the Farthest Point Sampling module which is used to sample points from a learnable latent high dimensional space, which has never been previously exploited. A combination of novel loss functions is used to reinforce the selection of points with increased curvature while at the same time retaining the overall structure of the point cloud. The proposed method not only achieves better perceptual similarity preservation compared to traditional method but also attain a better runtime. The simplification module shows great generalization and can be directly applied without further training to any point cloud. Finally, we showcase that such method can be apply to meshes simply by following a triangulation step after the sampling phase.

- In Chapter 6 we built upon the idea of learnable simplification method and we propose a neural mesh simplification framework. Although the point cloud simplification method can sample salient points with high performance, it lacks the ability to directly optimize the surface of the simplified mesh. This limits its applicability in mesh simplification scenarios. To tackle this limitation and also accelerate the runtime of the point sampling module we propose a GNN-based neural mesh simplification module that learns not only to sample points from the input mesh but also triangulate them. Compared to traditional simplification approaches that collapse edges in a greedy iterative manner, we propose a fast and scalable method that simplifies a given mesh in one-pass. The contributions of this Chapter are three-fold. Initially, we develop a constant complexity GNN-based sampling module that extends the notion of random sampling by sampling from a multinomial distribution. To train this module we propose a soft relaxation of Chamfer distance that assigns

high inclusion scores to points that preserve the structure of the input. Secondly, we introduce a learnable triangulation module that predicts candidate triangles and assigns an inclusion score according to their properties. To formulate this module, we propose to construct a graph on the candidate faces to enable information exchange between them and then apply GNN layers to the face graph. We demonstrate that such formulation is crucial for one-pass triangulation and the proposal of triangles that respect manifold properties. Thirdly, we introduce a set of novel loss function that aim to preserve both the appearance and the structure of the input mesh along with the surface properties of a mesh. Experimentally we show that those loss function enable the generation of smooth, manifold and watertight simplified meshes that outperform the traditional state-of-the-art methods. The method proposed in this Chapter is lightweight and fully differentiable which can be translated in large flexibility in the selection of loss functions and direct adaptability to any learnable pipeline without a significant overhead. Finally, we demonstrate that the running performance can be up to 10-times faster than traditional methods.

1.4 Impact and Applications

In this thesis we exploit the impact of geometric deep learning in common 3D task such as modeling and simplification. The proposed hand model, the dynamic face 3DMM along with the point cloud and mesh simplification methods that were developed through this thesis can be used in a large variety of application including but not limited to:

- **Medical and Human Assistive Applications**: Given that the proposed hand model is photorealistic and highly detailed it can aid the development of realistic and practical prosthetics for hand disabled people. Additionally, it can impact the creation of speech-to-sign-language methods that will translate speech signals to sign language for deaf persons. Finally, the large amount of available subjects' meta-data can contribute to biometric applications in estimating the weight and the height along with the ethnicity of a subject based on their hands.
- **Graphics**: Undoubtedly, a generative model that is able to animate a human face in a photorealistic manner would be beneficial in the gaming and graphics community along with augmented and virtual reality research (AR/VR). The task of human avatar animation in a human-like way is an ongoing task for the gaming industry and filming industry with current state-of-the-art role-play games being far away from realism. In addition, our simplification methods can aid the graphics community

to accelerate rendering, which is the biggest bottleneck in real-time applications, by rendering simplified meshes that preserve the details of the original.

- **Autonomous Driving**: Nowadays, LiDAR (Light Detection and Ranging) sensors are one of the key technologies used in autonomous driving cars to enable them to perceive and interact with their environment. However, although LiDAR sensor can scan 3D scenes in real-time they generate extremely dense point clouds containing an enormous amount of redundant information creating a computational burden in real-time processing. The point cloud simplification method proposed in this thesis could be proven beneficial for the detection of semantic features of the scans and aid the performance of autonomous driving systems.

- **Scanning devices**: With the advent of 3D scanning devices it is now possible to acquire highly detailed 3D meshes and point clouds with high frames per second (FPS) ratios. Nevertheless, such meshes and point clouds usually come at an unnecessary increased resolution that leads to huge storage requirements. Using the proposed simplification methods one could simplify the acquired scans to the required resolution as well as reinforce the simplification objective according to the desired properties of the object.

1.5 Publications

In this Section, a list of publications is provided that I authored during the years of my PhD studies. The publication list is split in two parts: a) the publications which are the direct outcome of this thesis' objectives, and b) the publications that are not directly related and will not be covered in detail.

1.5.1 Relevant Publications

- **Rolandos Alexandros Potamias**, Jiali Zheng, Stylianos Ploumpis, Giorgos Bouritsas, Evangelos Ververas, Stefanos Zafeiriou, Learning to Generate Customized Dynamic 3D Facial Expressions, Proceedings of the European Conference on Computer Vision (ECCV), 2020 [23].

- **Rolandos Alexandros Potamias**, Giorgos Bouritsas, Stefanos Zafeiriou, Revisiting point cloud simplification: A learnable feature preserving approach, Proceedings of the European Conference on Computer Vision (ECCV), 2022 [10].

- **Rolandos Alexandros Potamias**, Stylianos Ploumpis, Stefanos Zafeiriou, Neural Mesh Simplification, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022 [24].
- **Rolandos Alexandros Potamias**, Stylianos Ploumpis, Stylianos Moschoglou, Vasilios Triantafyllou, Stefanos Zafeiriou, Handy: Towards a high fidelity 3D hand shape and appearance model, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023 [25].

1.5.2 Other Publications

- **Rolandos Alexandros Potamias**, Alexandros Neofytou, Kyriaki Margarita Bintsi, Stefanos Zafeiriou, Graphwalks: efficient shape agnostic geodesic shortest path estimation, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR-w), 2022 [26].
- Kyriaki-Margarita Bintsi, Vasileios Baltatzis, **Rolandos Alexandros Potamias**, Alexander Hammers, Daniel Rueckert, Multimodal brain age estimation using interpretable adaptive population-graph learning, Medical Image Computing and Computer Assisted Intervention (MICCAI), 2023 [27].

1.6 Thesis Outline

In Chapter 2 we introduce the essential background knowledge on geometric deep learning and graph neural networks and we present the current state-of-the-art models. Chapter 3 presents a static neural morphable model for the case of hand shape modeling and GAN-based texture model. In Chapter 4 we present the first neural dynamic morphable model that is able to generate realist 4D facial animations. In Chapter 5 we introduce the necessity for simplification on 3D models and we present a GNN method to simplify point clouds. The task of simplification is extended in meshes in Chapter 6, where we present a real-time neural mesh simplification method. Finally, in Chapter 7, we conclude the findings of this thesis and we present several future directions of research.

CHAPTER 2

BACKGROUND AND RELATED WORK

Contents

2.1 Preliminaries

2.1.1 Principal Component Analysis

Principal Component Analysis (PCA) is a widely used dimensionality reduction technique in data analysis and machine learning. It is used to identify the most important features or patterns in a dataset and transform the data into a new coordinate system called the principal components.

The goal of PCA is to represent a high-dimensional dataset in a lower-dimensional space while retaining as much of the original information as possible. This is achieved by finding a set of orthogonal axes, called principal components, where the first principal components captures the maximum variance in the data. Essentially, by projecting the data onto the principal components, the maximum variance between them will be retained which is able to capture most of the information of the data.

In a mathematical setting, given a dataset with N data samples $[\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, ..., \mathbf{x_N}]$, each consisting of D features, we can represent the dataset as a matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$. The dataset is assumed to be centered with zero mean, or a normalization step is usually performed

to center the dataset around the axis origin. The first step of the decomposition requires the calculation of the covariance matrix of X:

$$\mathbf{S} = Cov(X) = \frac{1}{N}\mathbf{X}\mathbf{X}^T \tag{2.1}$$

PCA aim to identify an orthogonal subspace of rank $D' << D$, that maximizes the variance in the dataset. This can be mathematically formulated as an orthogonal subspace $\mathbf{W} = [\mathbf{w}_0, \mathbf{w}_0, ..., \mathbf{w}_{D'}] \in \mathbb{R}^{D \times D'}$ that is able to transform the data samples $\mathbf{x}_i$ to a lower rank projection $\mathbf{y}_i = \mathbf{W}^T\mathbf{x}_i$, where $\mathbf{y}_i \in \mathbb{R}^{D'}$.

The optimal subspace $\mathbf{W}$ can be identified by solving the system:

$$\begin{aligned} &\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmax}}[\operatorname{trace}(\mathbf{W}^T\mathbf{S}\mathbf{W})] \\ &\text{subject to } \mathbf{W}^T\mathbf{W} = \mathbf{I} \end{aligned} \tag{2.2}$$

The solution of the aforemention system is given by obtaining the Lagrangian:

$$\text{Lagrangian}(\mathbf{W}, \lambda) = \mathbf{W}^T\mathbf{S}\mathbf{W} - \lambda(\mathbf{I} - \mathbf{W}^T\mathbf{W}) \tag{2.3}$$

which results to the equation of eigenvalues of $\mathbf{W}$:

$$\mathbf{S}\mathbf{W} = \mathbf{\Lambda}\mathbf{W} \tag{2.4}$$

By selecting the appropriate number k of eigenvectors $\mathbf{v}_i$ corresponding to the k largest eigenvalues we are able to obtain a projection matrix $\mathbf{W}$ that reduces the dimensionality of the data while maximizing their variance.

Principal Component Analysis, given its ability to effectively reduce the dimensionality of datasets, is a widely used technique with application ranging from feature extraction and noise reduction to 3D shape modeling.

2.1.2 Graph Definition

A graph $\mathcal{G}$ is defined as a set $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ of a finite number of N vertices $\mathcal{V}$ that are connected through the edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and weights $\mathbf{W}$. More specifically, two vertices i, j are connected with weight $w_{i,j} \in \mathbf{W}$ if there is an instance (i, j) in the set of edges

$\mathcal{E}$. From the set of edges $\mathcal{E}$, one can construct the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N,N}$ that is filled with non-zero values if and only if two vertices i and j are connected with an edge $e_{i,j} \in \mathcal{E}$. In such case, the value $a_{i,j} = 1$. If the graph is undirectional the matrix $\mathbf{A}$ is symmetric. In contrast, in the case of a directional graph, a non-zero entity $a_{i,j}$ on the adjacency matrix does not imply a non-zero entity $a_{j,i}$ as well. Throughout this thesis only undirectional graphs are considered. Each graph is also associated with a diagonal degree matrix $\mathbf{D}$, indicating the number of edges associated with each vertex.

2.1.3 Convolution

Convolution operator has been the key to success of deep learning models, achieving state-of-the-art performance in almost every grid structured problem. One of the key properties that contributed to its success is the preservation of the relationships between features and patterns when the input is translated. This property is named **translation equivariance** and enables convolutional layers to operate on local neighborhoods of the input data using shared weights, allowing them to detect similar patterns regardless of their spatial location in the input. The main idea behind convolutions is the stationarity of data to locally repeat similar features. Leveraging the stationarity property of the data, one can reduce the number of parameters required by using shared weights across local regions. Given that convolutions learn spatially invariant representations, meaning that the learned features are robust to translations or shifts in the input data, they are extremely efficient in tasks where the location of objects or features of interest within the input can vary, and it is important for the model to recognize them regardless of their position. In the Euclidean setting, the convolution between two continuous functions at point t can be defined as:

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau \tag{2.5}$$

where f, g are two functions and $*$ the convolution operator.

Similarly, in the discrete setting, convolution of a discrete point n can be defined as:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] g[n - m] \tag{2.6}$$

where f, g two discrete functions.

In the special case of finite signals f, g, a periodical wrapping is applied, usually called cyclic or periodic convolution. It is called "cyclic" because it treats the input sequences as periodic signals that repeat indefinitely. The two input sequences are assumed to be periodic, with the period equal to their respective lengths. The resulting sequence is also of the same length as the input sequences. This operation is usually formulated by transforming the filter function g in Eq. 2.8, to a circulant matrix $C(g)$:

$$C(g) = \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_{m-1} \\ g_{m-1} & g_0 & g_1 & \cdots & g_{m-2} \\ g_{m-2} & g_{m-1} & g_0 & \cdots & g_{m-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & g_3 & \cdots & g_0 \end{bmatrix} \tag{2.7}$$

Using the circulant matrix formulation, the convolution operator can be redefined as a simple matrix-vector multiplication:

$$(f * g) = C(g)f \tag{2.8}$$

An important derivative of circulant matrices is their commutativity property with the shift operator. This also leads to the shift equivariance of convolution operator.

However, the notion of translation $g[n - m]$ introduced Eq. 2.8 by shifting a point n by m, is elusive in the case of unstructured and irregular domains. Several approaches have attempted to redefine the operation of convolution to non-Euclidean structures under the umbrella term of geometric deep learning.

2.2 Geometric Deep Learning

Recently, the enormous amount of applications related to data residing in non-Euclidean domains motivated the need for the generalization of several popular deep learning operations, such as convolution, to graphs and manifolds. The main efforts include the reformulation of regular convolution operators in order to be applied on structures that lack consistent ordering or directions, as well as the invention of pooling techniques for graph downsampling. All relevant endeavours lie within the new research area of Geometric Deep Learning (GDL) [4].

2.2.1 Spectral Domain

As discussed previously, in the non-Euclidean case we cannot even define the translation operation $n-m$, as in Eq. 2.8, on the manifold or graph, so the notion of convolution does not directly extend to this case. An alternative definition of convolution in the frequency domain can arise using the diagonalization of circulant matrices.

In particular, all commuting matrices, such as circulant matrices as defined in Eq. 2.7, can be jointly diagonalized. Such property entails that all circulant matrices share the same eigenvectors that diagonalize them. In addition, those eigenvectors will then simultaneously give the diagonalizing transformation $\mathbf{\Phi}^{-1}C(g)\mathbf{\Phi}$, were the columns of matrix $\mathbf{\Phi}$ will be the eigenvectors of the circulant matrix. It can be easily proven using the shift operator as a simple case of the circulant matrix that its eigenvectors correspond to the Fourier basis. Given the joint diagonalization property of circulant matrices this can be extended to any arbitrary circulant matrix. Thus, the convolution can be diagonalized by the Fourier basis as:

$$(\mathbf{f} * \mathbf{g}) = \mathbf{\Phi}\,\mathrm{diag}(\hat{g})\mathbf{\Phi}^T\mathbf{f} \tag{2.9}$$

where $\mathrm{diag}(\hat{g})$ is a diagonal matrix containing the eigenvalues of the circulant matrix $C(g)$, which also represent the Fourier transform of the values of $C(g)$. The main motivation behind the decomposition of convolution operator to the Fourier basis is the popular Fourier transform duality, in which a convolution between two signals in the spatial domain can be calculated using matrix multiplication in the frequency domain. Indeed, Eq. 2.9, can be reformulated using the previous definitions to:

$$(\mathbf{f} * \mathbf{g}) = \underbrace{\mathbf{\Phi}\underbrace{\mathrm{diag}(\hat{g})}_{\mathbf{G}=\mathcal{F}(g)}\underbrace{\mathbf{\Phi}^T\mathbf{f}}_{\mathbf{F}=\mathcal{F}(f)}}_{\mathcal{F}^{-1}(\mathbf{GF})} \tag{2.10}$$

where $\mathcal{F}$ represents the Fourier operator and $\mathcal{F}^{-1}$ its inverse.

One of the most popular circulant matrices on discrete domains is the **graph Laplacian**, given its ability to be defined on diverse structures. The Laplacian operator, for an n-dimensional signal $f : \mathbb{R}^n \to \mathbb{R}$, can be defined as a combination of the intrinsic gradient and divergence operators :

$$\mathcal{L}_f = \Delta f = -div(\nabla f) = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2} \tag{2.11}$$

In the discrete case of graphs, the Laplacian operator quantifies how a function defined on the graph changes with respect to its neighboring nodes:

$$\Delta f(i) = \sum_{j \in \mathcal{N}_i} w_{i,j}[f(i) - f(j)] \tag{2.12}$$

where $f(i)$ is the function f applied to node i and $\mathcal{N}_i$ is the set of its neighbours.

This can be reformulated using the degree $\mathbf{D}$ and the weight matrix $\mathbf{W}$:

$$\mathbf{\Delta} = \mathbf{D} - \mathbf{W} \tag{2.13}$$

In the special case of an unweighted graph, Eq. 2.13 transforms to $\Delta = \mathbf{D} - \mathbf{A}$, where $\mathbf{A}$ is the graph adjacency matrix. However, in most cases the normalized version of the Laplacian is being used:

$$\Delta f = \mathbf{I}_n - D^{-1/2} W D^{-1/2} \tag{2.14}$$

with $\mathbf{I}_N$ defines the $N \times N$ identity matrix. Apart from the circulant matrix eigenvalues, one can prove the relationship between Fourier basis and Laplacian eigenvalues using Dirilecht energy function [4] or by expanding the classical Fourier definition for continuous functions [28].

Similar to the circulant matrices, the Laplacian matrix $\mathbf{\Delta}$ is a real symmetric positive semi-definite matrix with a complete set of orthonormal eigenvectors $\{\phi_i\}_{i=0}^{N}$, known as the graph Fourier basis and their corresponding eigenvalues $\{\lambda_i\}_{i=0}^{N}$ identify as frequencies of the graph. Using the matrix of Fourier basis $\mathbf{\Phi} \in \mathbb{R}^{N \times N}$ and the diagonal matrix $\mathbf{\Lambda} = \text{diag}(\lambda_0, \lambda_1, ..., \lambda_N) \in \mathbb{R}^{N \times N}$ of the eigenvalues we can diagonalize the Laplacian matrix as $\mathbf{\Delta} = \mathbf{\Phi \Lambda \Phi}^T$. Using the above formulation, it follows that a signal $\mathbf{f}$ defined on the graph can be filtered by a function g_θ on the spectral domain as:

$$\mathbf{f}' = g_\theta(\mathbf{\Delta})\mathbf{f} = g_\theta(\mathbf{\Phi \Lambda \Phi}^T) = \mathbf{\Phi} g_\theta(\mathbf{\Lambda}) \mathbf{\Phi}^T \mathbf{f} \tag{2.15}$$

Leveraging the definition of convolution using the Fourier basis (Eq. 2.9), Bruna *et.al.*[29] defined the first non-parametric spectral convolution layer (**SpectralCNN**), acting on irregular domains as:

$$\mathbf{f}_j = \sigma\left(\sum_i \mathbf{\Phi}\mathbf{G}_{\mathbf{i,j}}\mathbf{\Phi}^{\mathbf{T}}\mathbf{f}_{\mathbf{i}}\right) \tag{2.16}$$

where $\mathbf{f}_j$ the features of node j, $\mathbf{\Phi}$ the eigenvectors of the graph Laplacian Δ, $\mathbf{G}_{\mathbf{i,j}}$ the trainable diagonal matrix of spectral multipliers and $\sigma(\cdot)$ a non-linearity applied vertex-wise to the output of the convolution. To model the non-parametric filter $\mathbf{G}$, the authors utilized a cubic B-spline basis. Additionally, the authors proposed to retain only the first k eigenvectors of the Laplacian given that they capture most of the smooth details of the graph. However, this approach comes with several drawbacks. In particular, it is not scalable due to the requirement of multiplying the data by the graph Fourier basis $\mathbf{\Phi}$. While the computation cost of this matrix is not neglectable ($\mathcal{O}(N^2)$), the primary bottleneck is the need to perform two multiplications (forward and inverse Fourier transforms) on the data, resulting in a computational complexity of operations per forward and backward pass. Also, since their model relies on smoothness in the Fourier domain and uses spline parametrization to introduce localization in the vertex domain, it lacks precise control over the local support of their kernels, which is crucial for learning filters with specific localization. Finally, given the non-parametric form of the filter $\mathbf{G}$, the number of parameters of the layer depend linearly on the input which deviates from the design of a shared constant number of parameters that traditional Euclidean convolution have.

Deferrard *et.al.*[30] proposed ChebNet, to address the limitations of SpectralCNN [29], by applying a filter parametrisation based on polynomials of the eigenvalues of the graph Laplacian. In particular, the authors defined the spectral filter $g_\theta(\Lambda)$ as:

$$g_\theta(\mathbf{\Lambda}) = \sum_{k=0}^{K-1} \theta_k \mathbf{\Lambda}^k \tag{2.17}$$

where $\theta \in \mathbb{R}^K$ a learnable vector of coefficients. The corresponding convolution can be now defined as:

$$
\begin{aligned}
\mathbf{f} *_{\mathcal{G}} \mathbf{g}_\theta &= \mathbf{\Phi} g_\theta(\mathbf{\Lambda}) \mathbf{\Phi}^T \mathbf{f} \\
&= \mathbf{\Phi} \left(\sum_{k=0}^{K-1} \theta_k \mathbf{\Lambda}^k \right) \mathbf{\Phi}^T \mathbf{f} \\
&= \left(\sum_{k=0}^{K-1} \theta_k \mathbf{\Phi} \mathbf{\Lambda}^k \mathbf{\Phi}^T \right) \mathbf{f} \\
&= \left(\sum_{k=0}^{K-1} \theta_k \mathbf{\Delta}^k \right) \mathbf{f} \\
&= \hat{g}_\theta(\mathbf{\Delta}) \mathbf{f}
\end{aligned}
\tag{2.18}
$$

Using this formulation, the number of parameters of the filter are independent from the input size and are only related to the degree of the polynomial. To tackle this, a parametrization of $g_\theta(\mathbf{\Lambda})$ as a polynomial function that can be computed recursively from the graph Laplacian, given that K multiplications by a sparse Laplacian costs $O(|\mathcal{E}|) << O(|N^2|)$ One such polynomial, traditionally used in Graph Signal Processing to approximate kernels (like wavelets), is the Chebysev polynomial [31]. Recall that the Chebysev polynomial $T_k(x)$ of order k can be evaluated as:

$$
\begin{aligned}
T_0(x) &= 1 \\
T_1(x) &= x \\
T_k(x) &= 2xT_{k-1}(x) - T_{k-2}(x)
\end{aligned}
\tag{2.19}
$$

The convolution filter can be parametrized as truncated expansion with Chebysev polynomial of order K, where θ_k are learnable Chebysev coefficients. Using this parametrization, the number of convolution parameters are independent of the size of the graph, which enables the scalability of the operator. Furthermore, another important property of this formulation is that the filters are localized in the spatial domain given that both the Laplacian along with its powers $\mathbf{\Delta}^k$ are local operators acting around a k-hop neighborhood.

In a following work, Kipf and Welling [32], proposed **Graph Convolutional Network (GCN)** to simplify the Chebysev polynomial to a linear form, i.e. k = 2 which results to 1-hop neighborhood localized filters with only two parameters for each convolution. The reparametrized convolutions, using the normalized Laplacian, were defined as:

$$
\begin{aligned}
\mathbf{f} *_{\mathcal{G}} \mathbf{g}_\theta &= \hat{g}_\theta(\mathbf{\Delta}) \mathbf{f} \\
&= \theta_0 \mathbf{f} - \theta_1 \Big(\mathbf{D^{-1/2} W D^{-1/2}} \Big) \mathbf{f}
\end{aligned}
\tag{2.20}
$$

where θ_0, θ_1 are the parameters of the convolution. By setting $\theta_0 = -\theta_1$ the graph convolution can be further constrained to one-parameter $\mathbf{f} *_{\mathcal{G}} \mathbf{g}_\theta = \theta_0\left(\hat{\mathbf{D}}^{-1/2}\hat{\mathbf{W}}\hat{\mathbf{D}}^{-1/2}\right)\mathbf{f}$, where $\hat{\mathbf{W}} = \mathbf{W} + \mathbf{I}$ and $\hat{\mathbf{D}} = \text{diag}(\sum_{j \neq i} \hat{w}_{ij})$.

2.2.2 Spatial Domain

Spectral approaches that aim to generalize convolutions to irregular domains, suffer from the inherent drawback in their ability to generalize across different domains. As previously discussed, spectral convolutions rely on the Fourier basis which is domain dependent, meaning that if the domain slightly changes the Fourier basis will be totally different. To enable cross domain generalization several approaches have been developed that act directly on local charts and patches on the spatial space. As one can easily identify, both ChebNet and GCN methods boil down to applying simple filters acting on the k-hop neighborhood of the graph in the spatial domain. Spatial convolutions can be thought as the analogy of a kernel applied on a patch of a Euclidean signal, by locally aggregating information around its node. In their general form, spatial convolutions can be defined as:

$$\begin{aligned} (f * g)(\mathbf{v}_i) &= \sum_j g_j D_i(\mathbf{v}_j) F \\ D_i(\mathbf{v}_j) f &= \int f(\mathbf{v}_i) u(\mathbf{v}_j, \mathbf{v}_i) d\mathbf{v}_i \end{aligned} \tag{2.21}$$

where i, j two nodes of the graph, and f a function applied on them.

Masci*et.al.*introduced GeodesicCNN [5], convolution operator that leverages the intrinsic properties of the manifold by applying filters to local patches represented using geodesic polar coordinates. In particular, the polar coordinates of a node $\mathbf{v}_j$ can be given using the intrinsic distance $\rho(\mathbf{v}_j) = \text{dist}(\mathbf{v}_j, \mathbf{v}_i)$ and the angular coordinate $\psi(\mathbf{v}_j)$ as:

$$u(\mathbf{v}_i, \mathbf{v}_j) = e^{-(\psi_j - \psi_i)^2 / 2\sigma_\psi^2} e^{-(\rho_j - \rho_i)^2 / 2\sigma_\rho^2} \tag{2.22}$$

where ρ_j, ψ_j represent the distance and the angular coordinates of node j respectively.

The idea of GeodesicCNN to utilize non-isotropic kernels extended in AnisotropicCNN [33], by utilizing heat diffusion kernels to break the isotropic nature of the graph kernels. The differences between kernels on a local patch of a 3D mesh can be illustrated in Figure 2.1.

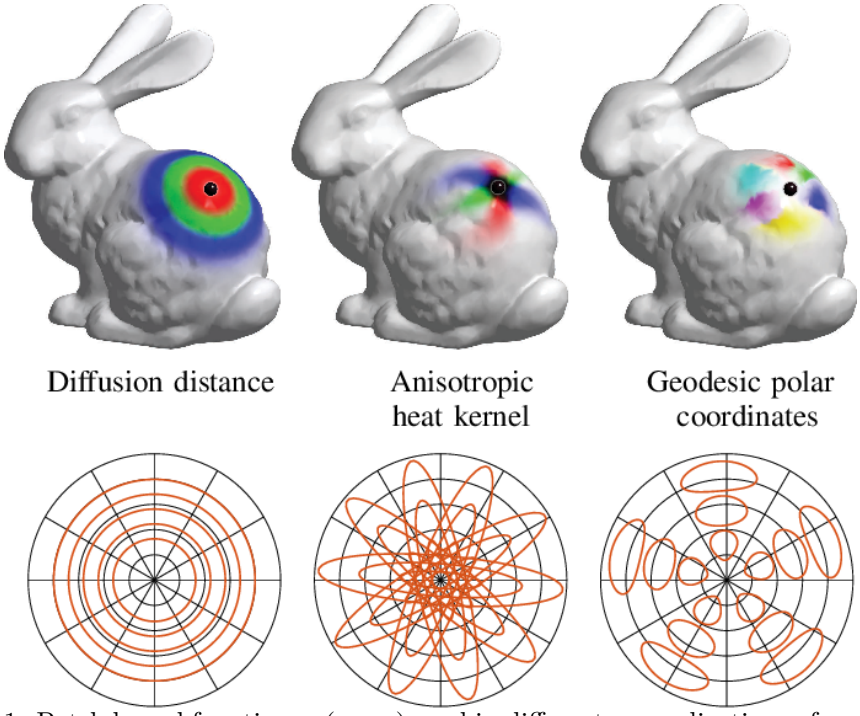


Figure 2.1: Patch kernel functions $\mathbf{u}(\mathbf{v}_i, \mathbf{v}_j)$ used in different generalizations of convolution on the manifold (Figure from [4])

MoNet [34] extend the kernel operator to handle both manifolds and graphs, making it adaptable through learning. They define each patch operator as a weighted average of the signal from neighboring points around a central point. The weights are determined by learnable functions that take as input the central point and one of its neighbors, and these functions are parameterized based on a pseudo-coordinate system. To increase the degrees of freedom of the convolution, the authors propose to learn not only the filters but also the patch operators by using a Gaussian kernel with learnable mean and covariance matrices:

$$u(\mathbf{v}_i, \mathbf{v}_j) = \exp\left(-\frac{1}{2}\left(\mathbf{v}_j - \mu_\mathrm{i}\right)^T \mathbf{\Sigma}_i^{-1}\left(\mathbf{v}_j - \mu_\mathrm{i}\right)\right) \tag{2.23}$$

where $\mu_\mathrm{i}, \mathbf{\Sigma}_i$ are the learned mean and the covariance matrices of patch i. It can be easily seen that all of the aforementioned graph convolution operators can be expressed using the MoNet formulation. Several follow-up works extended the notion of learned patch kernel by using b-spline kernels [35] or by learning the pseudo-coordinate transformation [36].

Undoubtedly, a generalization of the previous methods called **Graph Attention Networks (GAT)** revolutionized the field of graph learning. GAT inspired from the attention-based models [11], reformulated graph convolutions as message passing networks. In par-

ticular, for every node of the graph a self-attention mechanism is applied between the node and its neighbours:

$$a_{ij} = \text{softmax}_j \quad \alpha\left(\mathbf{F}(\mathbf{v}_i), \mathbf{F}(\mathbf{v}_j)\right) \tag{2.24}$$

where $\alpha(\cdot)$ is the self-attention mechanism and $F(\mathbf{v}_i)$ is a learnable function over the node i. The authors proposed to use a linear transform between the concatenated features of the two nodes followed by a non-linearity as self-attention function: $\alpha\left(\mathbf{F}(\mathbf{v}_i), \mathbf{F}(\mathbf{v}_j)\right) = \xi\left(\mathbf{a}^T[\mathbf{F}(\mathbf{v}_i)||\mathbf{F}(\mathbf{v}_j)]\right)$, where $||$ represents the concatenation operator and $\xi(\cdot)$ a non-linearity.

Following GAT, several works have been proposed to generalize machine learning to irregular domains such as graphs and manifolds [37, 38, 39] which can be generalized under the update rule of **Message Passing Neural Networks**:

$$\mathbf{v}_i' = \xi\left[\gamma_{\mathbf{\Theta}}\left(\mathbf{v}_i, \underset{j \in \mathcal{N}_i}{\square} \eta_{\mathbf{\Theta}}\left(\mathbf{v}_i, \mathbf{v}_j, \mathbf{e}_{i,j}\right)\right)\right] \tag{2.25}$$

where $\gamma_{\mathbf{\Theta}}(\cdot), \eta\mathbf{\Theta}(\cdot)$ are differentiable learnable functions with networks parameters $\mathbf{\Theta}$, $\mathbf{v}_i$ a node of the graph and $\mathbf{v}_j \in \mathcal{N}_i$ its neighbour, $\mathbf{e}_{i,j}$ the features of edge (i, j) , $\square$ denotes a differentiable, permutation invariant function, e.g., sum, mean, min, max or mul and $\xi(\cdot)$ a non-linearity. As shown in [40], all of the aforementioned graph convolutions can be expressed using Eq. 2.25.

2.2.3 Geometric Deep Learning on Meshes and Point Clouds.

As discussed in the previous section, Meshes and Point Clouds can be considered as irregular domains in the context of geometric data processing. In both cases, the irregularity arises from the lack of grid or uniform structure in the arrangement of vertices or points. This irregularity poses challenges in processing and analyzing the data, as traditional methods developed for regular grids or structures may not directly apply. Specialized algorithms and techniques, such as those in the field of geometric deep learning, are often used to handle these irregular domains and extract meaningful information from meshes and point clouds.

Traditionally, Point cloud processing techniques handled discrete surfaces as unstructured sets of points without inherent notions of intrinsic distances or connectivity. **PointNet** [20], a groundbreaking approach, introduces a point set processing layer that employs a 1x1 convolution shared among all points, followed by batch normalization and ReLU activation. The resulting local features are then aggregated using max pooling to generate a global representation of the surface. Despite its simplicity, PointNet has demonstrated impressive performance in 3D object classification and point cloud segmentation tasks,

remaining competitive with more recent methods. An extension of PointNet, named **PointNet++** [22], was proposed to capture local structures of the point cloud using a Message Passing Neural Network scheme. In contrast to PointNet, PointNet++ groups point progressively using a so-called *abstraction layer* that aggregates local regions. Each abstraction layer is composed by three components, a *Sampling Layer*, a *Grouping Layer* and a *Point Layer* which sample centre points, group them with their k-neighbours and process them using a mini-PointNet network, respectively. Using the Message Passing Neural Network parlance of Eq. 2.25 we can define "abstraction layer" as:

$$\mathbf{v}_i' = \gamma_{\mathbf{\Theta}} \left(\max_{j \in \mathcal{N}_i} h_{\mathbf{\Theta}}(\mathbf{v}_j, \mathbf{p}_j - \mathbf{p}_i) \right) \tag{2.26}$$

where p_i, p_j are the vertex i, j $xyz-$positions. The Message Passing formulation of PointNet++ achieved state-of-the-art performance across various Point Cloud tasks and pioneered the use of geometric deep learning on Point Sets. Recently, several approaches leveraged this formulation to enhance point learning tasks [41, 42, 43, 26].

In contrast to Point Clouds, learning on Meshes requires construction of anisotropic filters, that leverage the underlying structure of the manifold. However, in a *fixed topology* setting, such an ordering is beneficial so as to be able to keep track of the existing correspondences. Several methods, exploited the fixed topology of a mesh to define a fixed ordering of the vertices and refrain from using permutation invariant operators. In particular, Lim *et.al.*[6] proposed to order the vertices of a neighborhood using a spiral trajectory, as shown in Figure 2.2 In a follow-up work, Bouritsas *et.al.*[7], defined a Message Passing Neural Network that uses an anisotropic soft-attention on spiral trajectories [6]. In particular, given a vertex $\mathbf{v}_i \in \mathcal{V}$, the authors introduced a *k-ring* and a *k-disk* as:

$$\begin{aligned} \text{0-ring}(\mathbf{v}) &= \mathbf{v}, \\ \text{(k+1)-ring}(\mathbf{v}) &= \mathcal{N}\left(\text{(k+1)-ring}(\mathbf{v})\right) - \text{k-disk}(\mathbf{v}), \\ \text{k-disk}(\mathbf{v}) &= \bigcup_{i=0,\dots,k} \text{i-ring}(\mathbf{v}) \end{aligned} \tag{2.27}$$

where $\mathcal{N}(V)$ is the set of all vertices adjacent to at least one vertex in the set V.

Once the k-ring is defined, the spiral trajectory centered around vertex $\mathbf{v}$ can be defined as:

$$S(\mathbf{v}, k) = \{\text{0-ring}(\mathbf{v}), \text{1-ring}(\mathbf{v}), \dots, \text{k-ring}(\mathbf{v})\} \tag{2.28}$$

In order to be consistent across all vertices, the authors pad or truncate $S(\mathbf{v}, k)$ to a fixed length L. To fully define the spiral ordering, the authors selected the initial vertex of $S(\mathbf{v}, k)$ to be in the direction of the shortest geodesic distance between a static reference

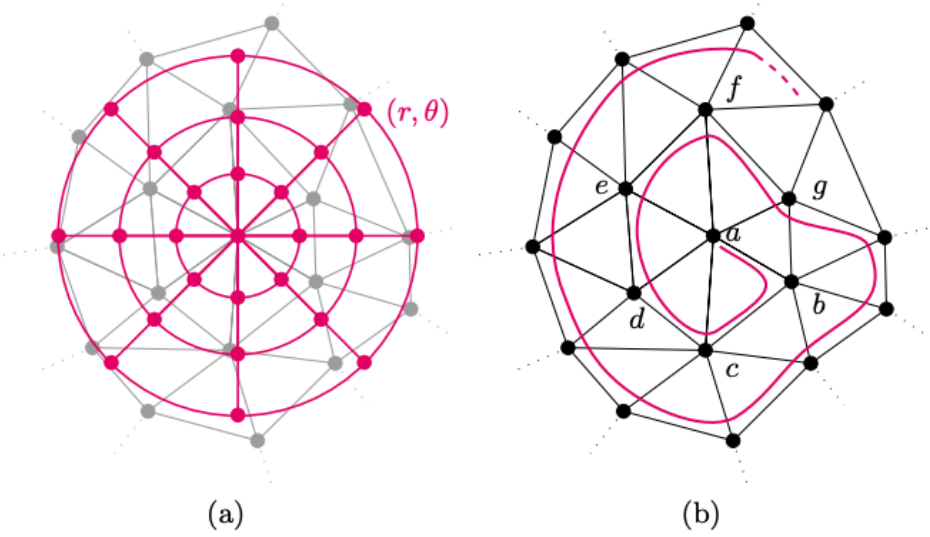


Figure 2.2: a) The polar coordinates constructed by GeodesicCNN [5] on a local patch b) The spiral trajectory that enumerates a fixed ordering of the neighbouring vertices of the patch. [6, 7]

vertex. Given that all 3D faces share the same topology, spiral ordering $S(\mathbf{v}, k)$ will be the same across all meshes and so, the calculation is done only once. With all the above mentioned, *Spiral Convolution* can be defined using the Message Passing Neural Network formulation as:

$$\mathbf{v}'_i = \gamma_{\mathbf{\Theta}} \left(\mathop{\Big\|}_{j \in S(\mathbf{v},k)} h_{\mathbf{\Theta}}(\mathbf{v}_j) \right) \tag{2.29}$$

where $\gamma_{\mathbf{\Theta}}$ and $h_{\mathbf{\Theta}}$ correspond to convolution learnable parameters, such as Multi-Layer Perceptrons (MLPs). Although both of the learnable functions could include a non-linear activation function, a non-linearity after the aggregation function $\gamma_{\mathbf{\Theta}}$ empirically achieved better performance.

The fixed ordering defined by the spiral trajectories in equivalence to traditional convolutions allows the use of long-studied practices in the computer vision community. For example, small patches can be used, leading to fewer parameters and fast computation. Furthermore, the authors showed that dilated convolutions can also be adopted in the spiral operator by simply subsampling the spiral.

2.3 Graph Pooling and Simplification

2.3.1 Graph Pooling

One of the components underlying the success of large-scale CNNs are pooling layers, introduced to formulate training in a hierarchical manner. Until recently, graph neural network (GNN) architectures used for tasks like classification, segmentation and generation, learn global graph representations by relying solely on node aggregations, neglecting the characteristics of local substructures. To mitigate such issues, several graph pooling layers have been introduced for hierarchical representation learning. Initial approaches, utilized variations of the non-trainable Graclus clustering algorithm [44, 45, 35] and Farthest Point Sampling [22] to perform pooling operations and generate hierarchical representations of the input node set. The first differentiable pooling layer (DiffPool) was introduced by [46] that learns a soft assignment matrix to perform node clustering. However, the clustering assignment matrix requires quadratic storage complexity and it is not scalable to large scale graphs [47]. To address the limitations of DiffPool several Top-K selection methods have been proposed, that select the top ranking nodes according to a learnable projection score [48, 47]. In order to enrich the projection score with local graph structure, SAGPool [49] utilized a GNN layer to assign self-attention scores to each node. However, Top-K approaches retain only a subset of the edge set of the input graph, leading to isolated nodes. Ranjan et al., [50] introduced ASAPooling, an extension to Top-k pooling schemes that performs node aggregation to address the edge connectivity limitations of the previous methods. Recently, a motif based pooling was introduced [51] that applied selection and clustering pooling techniques on the graph that was partitioned into small motifs.

2.3.2 Mesh Simplification

Mesh Simplification is a well studied field with long history of research. Traditional simplification algorithms repeatedly decimate the input mesh according to a cost function to preserve its rendered appearance, until the desired simplification ratio is reached. In an abstract sense, one may regard mesh simplification as a pooling process, since the input topology is given and its simplified version is unknown. However, in contrast to common graph pooling architectures, mesh simplification methods should also respect the surface properties of the mesh, such as smoothness and manifoldness. A natural approach to bypass the limitations is to attempt to bridge both words, as it will be described in Chapter 5 and Chapter 6.

Simplification methods can be distinguished in two major categories: *vertex clustering/-decimation* and *edge collapse* methods. Vertex decimation methods rank vertices according to a heuristic geometric cost function, such as their distance from the average plane [52, 53, 54], to ensure that least important vertices will be decimated first. However, although it is considerably more interpretable technique, a re-tessellation of the generated holes is required after each vertex deletion, making such algorithms impractical. On the other hand, edge collapse methods preserve the input topology by sequentially contracting pairs of vertices (i.e. edges). Hoppe *et.al.*[55, 56] pioneered an energy cost function defined over the edges that is attempted to be minimized in every contraction step. Following this idea, in the seminal works of [57, 58], each vertex was associated with the set of planes in its 1-ring neighborhood and was expressed by a fundamental *quadric* matrix. The authors showcased, that using the quadric matrix, the distance of a point from a set of planes can be expressed using the sum of their quadrics, which is known as Quadric Error Metric (QEM). Using this property, edges that introduce the minimum point-to-plane distance were the first to be collapsed. Several approaches have built upon QEM to incorporate texture [59, 60], curvature [61, 62, 63, 64], mesh saliency [65, 66, 67], spectral properties [68, 69], boundary constrains [60] or to speed-up the process using parallel processing [70, 71]. In [72] it was observed that greedy simplification methods lead to sub-optimal meshes and attempted to tackle mesh simplification as a global optimization problem using shape proxies. In particular, the authors introduced a normal deviation error metric to partition the input mesh to non-overlapping connected regions and then fit plane approximations (shape proxies) to each partition. Although the process produces more accurate shape approximations of the input, the method is not particularly efficient. Recently, Hanocka *et.al.*[73] proposed the utilization of an adaptive greedy edge collapse method as a learnable pooling strategy, where edge weights are learned through the network. However, apart from the inefficient greedy nature of the edge collapse methods, the resulting mesh faces can only be decimated approximately by a factor of two and thus limiting its applicability.

CHAPTER 3

STATIC NEURAL DEFORMABLE MODELS: THE CASE OF HAND MODEL

Contents

Over the last few years, with the advent of virtual and augmented reality, an enormous amount of research has been focused on modeling, tracking and reconstructing human hands. Given their power to express human behavior, hands have been a very important, but a challenging component of the human body. Currently, most of the state-of-the-art reconstruction and pose estimation methods rely on the low polygon MANO model [2]. Apart from its low polygon count, MANO model was trained with only 31 adult subjects, which not only limits its expressive power but also imposes unnecessary shape reconstruction constraints on pose estimation methods. Moreover, hand appearance remains almost unexplored and neglected from the majority of hand reconstruction methods. In this chapter, a large-scale model of the human hand is introduced and proposed, named "Handy++", which models both shape and appearance composed of more than 1200 subjects. The model is made publicly available for the benefit of the research community. In contrast to traditional models that are based on smooth PCA decomposition, the proposed model utilizes a mesh convolution operator that acts directly on the mesh space and learns hierarchical representations. In this way, semantically meaningful representations can be learned and the number of parameters can be considerably reduced. Additionally, the proposed hand model was trained on a dataset with

large diversity in age, gender, and ethnicity, which tackles the limitations of MANO and accurately reconstructs out-of-distribution samples. In order to create a high quality texture model, a powerful GAN is utilized, which preserves high frequency details and is able to generate high resolution hand textures. To showcase the capabilities of the proposed model, a synthetic dataset of textured hands was built and a hand pose estimation network was trained to reconstruct both the shape and appearance from single images. As it is demonstrated in an extensive series of quantitative as well as qualitative experiments, the proposed model proves to outperform the state-of-the-art and realistically captures the 3D hand shape and pose along with a high frequency detailed texture even in adverse "in-the-wild" conditions.

Figure 3.1: Our proposed hand model is able to generalise and accurately reconstruct the 3D hand shape and appearance from a single in-the-wild image. High frequency details are visible in the reconstructions such as wrinkles, veins, nail polish etc.

3.1 Introduction

Humans express their emotions mainly using their facial expressions and hands. Hand movements and poses are strong indicators of body language and can convey meaningful messages which can be key factors in human behavioral analysis. For this, hands have been widely studied in regard to their biometric applications [74, 75]. 3D hand models lead the technological developments of crucial tasks for virtual reality such as human hand tracking [76, 77, 78, 79] and pose estimation [80, 1]. Specifically, hand pose estimation algorithms utilize these models in order to reconstruct a subject's hand from a monocular depth or RGB image. However, most of the current state-of-the-art methods on 3D hand reconstruction and pose estimation rely on low polygon models, with minimum diversity in terms of age, gender, and ethnicity and without any hand texture appearance [2].

In particular, MANO [2] is considered the most popular hand model, which pioneered the construction of a parametric human hand model. Apart from its low polygon resolution

(778 vertices), it is composed by just 31 subjects, which limits the accuracy of high fidelity 3D reconstruction models. A statistical model with such a low number of samples will always constrain the reconstruction of hand shapes of diverse age and ethnicity groups. In the same context, despite the efforts of implementing strong pose priors to accurately constrain parametric models on valid hand poses [81], reconstruction methods are still dependent on a limited shape model. Importantly, current parametric models are constructed only by adults' hand shapes in the age range of 20-60 years old, disregarding the shape variations out of this age range. Experimental evidence shows that there is a significant difference in shape between children's and adults' hands. This difference makes current shape models more prone to reconstruction errors for arbitrary age gropus.

Additional to the shape component, a major limitation of current hand models is the absence of a high resolution texture model. Despite the necessity in virtual and augmented reality for a personalized appearance reconstruction, there are only a few studies that attempted to model hand texture along with shape and pose. In particular, current methods on hand texture reconstruction from monocular images are constrained on limited demographic variations and low resolution textures that are ill-suited for real-world applications [82, 83, 84, 8, 85]. Recently, HTML [8] proposed the largest available parametric texture model of the human hand composed of 51 subjects. Given that the texture component is based on Principal Component Analysis (PCA) of low resolution UV texture maps, the generated textures tend to be blurry, lacking the high frequency details of the hand. Low resolution textures not only limit the fidelity of RGB reconstructions but also the generations of realistic synthetic data. Currently, state-of-the-art hand-object detection methods [86, 87, 88] train their models on synthetic datasets with low resolution textures such as HTML or vertex colors, which subsequently constrain the quality of the resulting reconstructions.

In this study, the first large-scale parametric shape and texture hand model is proposed, named "Handy", composed of over 1200 subjects. Given these high resolution textured scans with large demographic, gender, and age variations, a high resolution hand model was built, which overcomes the shape limitations of previous state-of-the-art models. This is the first hand model that captures subjects with ages from 1 to 81 years old.

The scans come with high resolution textures which enable the creation of a highly detailed texture model. In contrast to HTML [8], a high resolution texture model was built, using a style-based GAN which allows the modeling of high frequency details of the human hand (e.g., wrinkles, veins, nail polish). Under a series of experiments, the proposed parametric model overcomes the limitations of previous methods and presents the first high fidelity texture reconstruction method from single "in-the-wild" images.

Besides the success of 3D hand reconstruction from monocular depth and RGB images, there are currently only a few methods that are able to reconstruct the pose along with the shape and texture components. Existing 3D hand datasets only contain hand annotations in terms of pose and global rotation and they usually neglect hand shape variations by modeling only a mean hand shape. Additionally, the lack of ground-truth high resolution texture maps limits current hand reconstruction methods to properly predict the appearance of a given hand. In order to enable texture modeling, the approach taken in this work follows the trend of synthetic data generation. A large-scale dataset was built, that encompasses annotations in terms of pose, shape, and texture information. In summary, the contributions of this Chapter are the following:

- A large-scale shape and appearance model of the human hand, built using over 1200 3D hand scans with a wide diversity in terms of age, gender, and ethnicity, which is made publicly available for the benefit of the research community.
- A synthetic dataset is created for monocular 3D hand reconstruction, utilizing the high fidelity hand model developed in this work. This dataset is made publicly available. As demonstrated in the experimental section, the synthetic dataset improves the performance of off-the-shelf reconstruction methods.
- A high fidelity appearance reconstruction method that is capable of reconstructing high frequency details such as wrinkles, veins, nail polish, and so on, from monocular images.

3.2 Related work

Parametric Hand Models

Over the years, several hand models have been proposed in the literature to approximate hand articulations. Initially, Oikonomidis *et.al.*[89] attempted to model hand shape as a collection of geometric primitives such as elliptic cylinders, ellipsoids, spheres, and cones. In the sequel [89], various approaches were proposed to model hand joints using anisotropic Gaussians [90], a collection of spherical meshes [91], or a union of convex bodies [92]. Schmidt *et.al.*[93] proposed the first implicit representation of the articulated hand using the popular Signed Distance Function. Khamis *et.al.*[94] proposed the first linear blend skinning (LBS) model constructed from 50 subject scans. The authors modeled hand poses and shape variations using a low dimensional PCA. To tackle the volume loss and

restrict unrealistic poses of the LBS, Romero *et.al.*[2] learned pose dependent corrective blend shapes from the scans of 31 subjects and proposed the MANO parametric model. Li *et.al.*[95] proposed the NIMBLE to model the interior of the hand, i.e. bones and muscles. Recently, HTML [8] attempted to create a parametric appearance model of the human hand by collecting hand textures from 50 subjects. However, given the limited amount of data, the authors train a PCA model on the UV space that results in low resolution textures. To address the aforementioned limitations, the present work proposes the first large-scale model of both hand shape and appearance of the human hand, composed of over 1200 scans.

Hand pose estimation

3D hand pose estimation has been a long studied field, originally tackled by deforming a hand model to volumetric [96] and depth images [97, 98, 99, 100]. Initially, 3D pose estimation was considered as a fitting problem where a 3D parametric model was used to fit 2D keypoints [101, 102]. De La Groce *et.al.*[83] pioneered hand pose tracking from single RGB images by solving an optimization problem. The advent of deep learning methods has shifted the research interest to sparse joint keypoints prediction from RGB images using CNNs [103, 80, 104, 102]. Most of these methods attempt to directly predict dense 3D hand positions by regressing the MANO model [2] parameters [87, 105, 106], which constrain them to the shape and pose space of MANO. Several methods try to deviate from MANO's parameter space, by directly regressing 3D vertex positions using graph neural networks [107, 108, 109]. Hasson *et.al.*[88] proposed a CNN-based method that regresses MANO and AtlasNet [110] parameters to reconstruct 3D hand poses together with various object shapes. Recently, a handful of methods attempted to reconstruct objects along with hands by using implicit [111, 112], parametric [113, 114, 86] or a combination of both representations [115].

Synthetic datasets for hand pose estimation

Synthetic datasets have been proven very effective, boosting training performance and overcoming data limitations in many applications, ranging from face reconstruction [116] to pedestrian detection [117]. Numerous amount of hand pose methods have been trained using synthetic data generated under different hand poses and illumination environments [103, 87, 118, 107, 86]. Hasson *et.al.*[88] rendered synthetic data using the SMLP model [119] under various hand poses from the GraspIt dataset [120]. Apart from the hands, the authors used objects from ShapeNet to generate a dataset of hand-object interactions.

However, all of the aforementioned studies are limited to only a few texture variations [103] or low resolution hand textures [84, 87, 88, 86], creating a domain gap between synthetic and real-world images. To address the domain gap, in this work a new dataset is created and proposed that consists both hands and objects. Similar to [88], the created dataset utilizes high-resolution textures of hands, taking a step towards a photorealistic synthetic hand dataset.

3.3 Handy++: Shape and Appearance Model

This section introduces the 3D dataset that was collected to build the high fidelity shape and texture model. Then, the process of bringing the entire hand dataset into dense correspondence and creating the large-scale shape model is described. Finally, the training of a style-based appearance model that preserves all the high frequency details of the human hand is detailed.

3.3.1 Large-scale 3D hand dataset

The large-scale 3D hand dataset used in this work was collected during a special exhibition at the Science Museum, London. A 3dMD structured light stereo system with 4 cameras was used to capture the hand data, producing high quality dense meshes. The raw scans have a resolution of approximately 30,000 vertices. A total of 1208 distinct subjects were captured, with available metadata including gender (53% male, 47% female), age ($1-81$ years old), height ($80-210$ cm), and ethnicity (82% White, 9% Asian, 7% Mixed and 2% black), as shown in Figure 3.2.

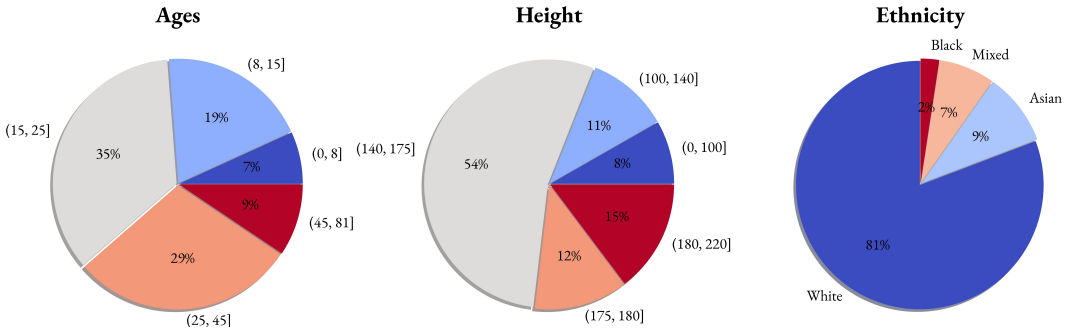


Figure 3.2: Distribution of demographic characteristics of the scanned subjects. The collected hand dataset covers a large variety of ages, heights, and ethnicities.

Most notably, the collected hand scans exhibit a large diversity in terms of age, ethnicity, and height, which provide a step towards a universal hand model. Compared to previous methods [2, 8], the scans collected include over 360 children aged less than 12 years old and 100 elderly subjects aged over 60 years old. In order to capture different pose variations, each subject was instructed to perform a range of hand movements according to a specific protocol each day for a period of 101 days. In particular, each subject was instructed to start from the open palm pose (canonical pose) and deform his hand according to several common poses and signs for 10 seconds, resulting in around 300 frames per subject. Each day a different pose protocol was utilized. Some example images can be seen in Figure 3.16. In this section, only the scans corresponding to the open palm pose utilized in order to construct a large scale hand shape model. Several samples of the collected dataset are illustrated in Figure 3.3.

Figure 3.3: Samples of different subjects under different poses on the collected dataset.

3.3.2 Raw data dense registration

To create a statistical shape model of the human hand, a set of 3D scans was rigidly aligned with a common template mesh. Two different resolution templates were used for this method. As a low polygon resolution template, the MANO template was utilized composed by 778 vertices, which can be directly adapted to the SMPL model [119]. For high quality hand modeling, a high resolution, in terms of polygons, hand template was utilized, comprising of 8407 vertices. The hand template was carefully designed by a graphics artist in order to include anatomical details of the hand such as veins and nails. A comparison between the MANO and the proposed template is shown in Figure 3.4.

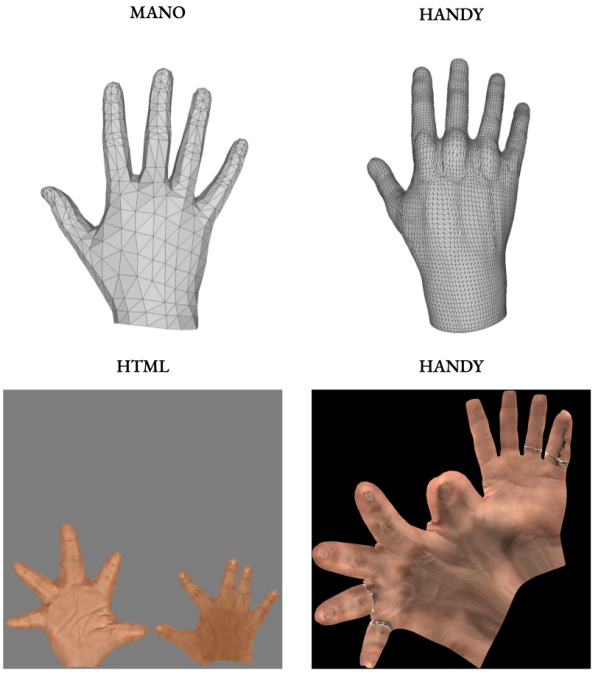


Figure 3.4: Samples of different subjects under different poses on the collected dataset.

To bring the raw scans into dense correspondence a five-step pipeline was used. Initially, the scans were rendered from multiple views and 2D joint locations were detected using MediaPipe framework [79]. Subsequently, the 2D joint locations were lifted to 3D by utilizing a linear triangulation and then the fingertips were detected using the projection of the finger skeleton to the tips of the surface. Using the 3D detected skeleton, a fitting process was performed by optimizing the pose parameters of a Linear Blend Skinning model (LBS) to align the template hand to the exact pose and shape of the raw scan. As an intermediate step of the raw scan registration, MANO [2] was used as an LBS model

to optimize pose θ and shape β parameters using following loss function:

$$\mathcal{L} = \mathcal{L}_J + \mathcal{L}_{col} + ||\beta||_2 + ||\theta||_2 \tag{3.1}$$

where $\mathcal{L}_J = ||J - \hat{J}||_2$ is a landmark loss that enforces MANO joints $\hat{J}$ to match the detected joints $\hat{J}$ and $\mathcal{L}_{col} = ||u_i - u_j||_2$ is a collision loss that applied to vertices v_i, v_j that penetrate the surface and enforces them to be in contact. To find the points that penetrate the surface we use the Winding Numbers algorithm. The optimization process was performed using Adam optimizer with learning rate of $1e-3$.

To obtain the fittings of the high resolution template, a manual mapping between the barycentric coordinates of the MANO and the high resolution Handy template was defined that was utilized to transfer the fitted MANO hand to the Handy template.

To acquire the hand dense registrations, Non-rigid Iterative Closest Point algorithm (NICP) [121] was applied between the fitted hand template mesh and the 3D raw scans. Finally, in order to avoid capturing any unnecessary deformations into the final shape model, a normalization step was performed that reposes registered hands to the canonical open-palm pose.

3.3.3 Handy: A PCA approach

As a baseline model, a deformable hand shape model described as a linear basis of shapes, was used. In particular, using PCA, a hand model was build with N vertices that is described by an orthonormal basis, after keeping the first n_c principal components $\mathbf{U} \in \mathbb{R}^{3N \times n_c}$ and their associated λ eigenvalues. This enabled the generation of hand instances by regressing the shape parameters $\boldsymbol{\beta} = [\beta_0, \beta_1, ..., \beta_{n_c}] \in \mathbb{R}^{n_c}$ as:

$$B_s(\boldsymbol{\beta}) = \mathbf{T} + \sum_{i=0}^{n_c} \mathbf{U_i}\beta_\mathbf{i} \in \mathbb{R}^{3N} \tag{3.2}$$

where $\mathbf{T} \in \mathbb{R}^{3N}$ refers to the mean hand shape. Variations of the first 5 shape components are illustrated in Figure 3.5.

Finally, the articulated hand model can be defined as:

$$\mathcal{M}(\boldsymbol{\beta}, \boldsymbol{\theta}) = W(T_p(\boldsymbol{\beta}, \boldsymbol{\theta}), J(\boldsymbol{\beta}), \boldsymbol{\theta}, \mathcal{W}) \tag{3.3}$$

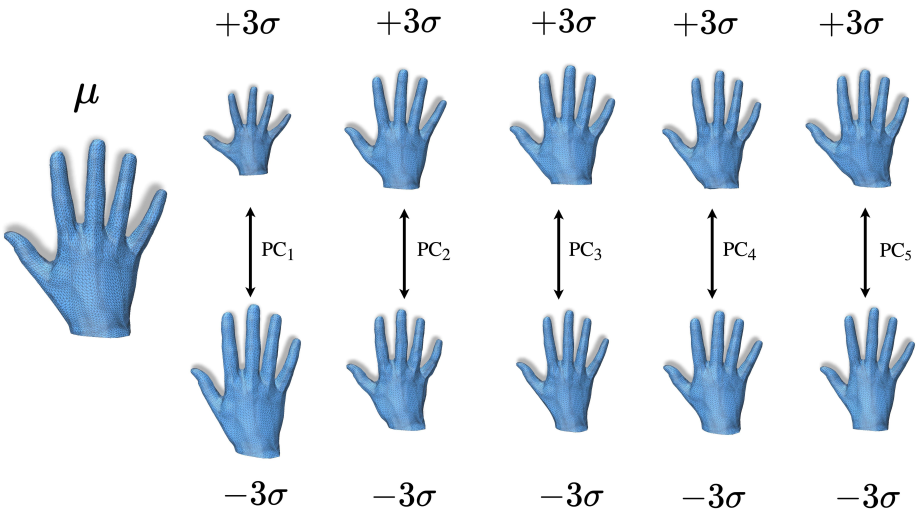


Figure 3.5: Mean shape **T** and the first five principal components, each visualized as additions and subtractions of 3 standard deviations ($\pm 3\sigma$) away from the mean shape.

$$T_p(\boldsymbol{\beta}, \boldsymbol{\theta}) = B_s(\boldsymbol{\beta}) + B_p(\boldsymbol{\theta}) \tag{3.4}$$

where $W(\cdot)$ corresponds to a linear blend skinning function (LBS) that is applied to the articulated hand mesh with posed shape T_p, $\mathcal{W}$, J define the blend weights and kinematic tree of joint locations, respectively, and $\boldsymbol{\beta}, \boldsymbol{\theta}$, are the shape and pose parameters, respectively. To tackle the joint collapse of typical LBS function, the MANO learned pose corrective blendshapes **P** [2] were used to produce more realistic posed hands:

$$B_p(\boldsymbol{\theta}) = \sum_{i=0}^{9K} \left(R_i(\boldsymbol{\theta}) - R_i(\boldsymbol{\theta}^*)\right) \mathbf{P}_i \tag{3.5}$$

where $\mathbf{P}_i$ are the pose blend shapes, K is the number of joints of the hand model, $R_i(\boldsymbol{\theta})$ is a function that maps pose parameters $\boldsymbol{\theta}$ to the rotation matrix of joint i and $\boldsymbol{\theta}^*$ refers to the canonical pose.

3.3.4 Handy++: a neural deformable hand model

Traditional shape modeling using a linear PCA is usually constrained given its limitation to learn high frequency details using compact representations. In this section, a neural

autoencoder is proposed to extend the "Handy" model. As shown in the literature, neural networks that utilize the structure of autoencoders can accurately model 3D faces and learn non-linear representations [122, 7]. Additionally, apart from the reconstruction quality, non linear neural-based methods have shown an extreme capability in learning up to 50% more compact representations compared to linear PCA models. The proposed neural deformable model utilizes spiral mesh convolutions as a building block to model hand shapes. As explained and discussed in Chapter 2, spiral mesh convolution, an ordering based graph neural network for fixed topology meshes, manages to achieve state-of-the-art performance by enforcing an explicit ordering of the neighbours of each vertex. Such ordering allows a "1-1" mapping between the neighbours and the parameters of a learnable local filter, similar to traditional convolution operator in Euclidean grids. Figure 3.6 shows an example of a spiral trajectory for a random vertex in the hand palm.

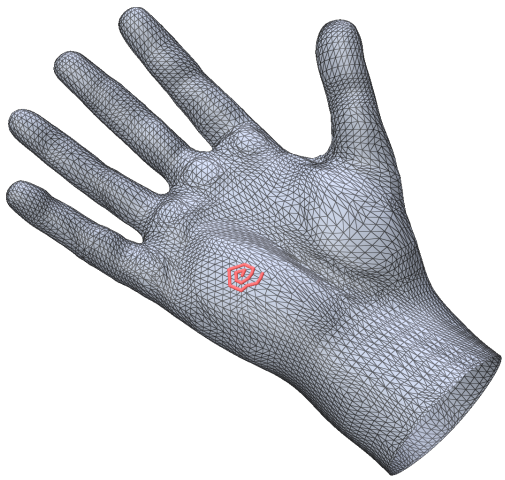


Figure 3.6: Example of a spiral trajectory around a vertex in the palm of the hand, that defines the ordering of its neighbors.

In essence, "Handy++" is a deep convolutional mesh autoencoder, that learns hierarchical representations of a shape. An illustration of the architecture is shown in Figure 3.7. Leveraging the connectivity of the graph with spiral convolutional filters, local processing of each shape is enabled. Furthermore, to enable hierarchical learning and allow learning in multiple scales, several pooling and unpooling layers are utilized similar to [122]. Both pooling and un-pooling operators are pre-defined sparse matrices obtained from quadric mesh simplification algorithm. The unpooling operators are based on sparse matrix multiplications with upsampling matrices $Q_u \in \{0,1\}^{n \times m}$, where $m > n$. Since upsampling operation changes the topology of the mesh, and in order to retain the face structure, upsampling matrices Q_u are defined on the basis of down-sampling matrices. To achieve

this, barycentric coordinates of the vertices that were discarded during downsampling procedure are stored and used as the new vertex coordinates of the upsampling matrices. This way, semantically meaningful representations can be learned while at the same time the number of parameters are considerably reduced. "Handy++" was trained using L_1 loss on the reconstructed hand meshes. The exact implementation details of the "Handy++" network architecture can be found in Table 3.1.

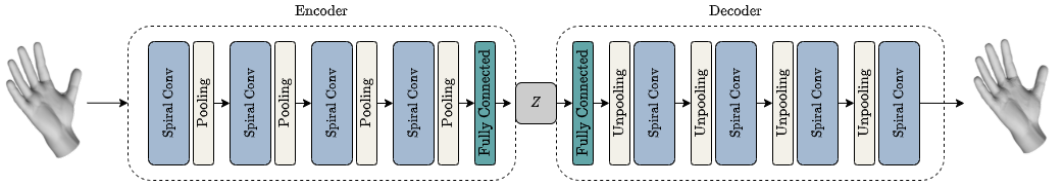


Figure 3.7: Overview of "Handy++", the proposed spiral autoencoder architecture.

Table 3.1: Implementation details of the "Handy++" architecture.

	Encoder Module			**Decoder Module**	
Layer	Input Dimension	Output Dimension	Layer	Input Dimension	Output Dimension
Convolution	28431x3	28431x8	Fully Connected	8	46x64
Downsampling	28431x8	5687x8	Upsampling	46x64	228x64
Convolution	5687x8	5687x16	Convolution	228x64	228x32
Downsampling	5687x16	1138x16	Upsampling	228x32	1138x32
Convolution	1138x16	1138x32	Convolution	1138x32	1138x16
Downsampling	1138x32	228x32	Upsampling	1138x16	5687x16
Convolution	228x32	228x64	Convolution	5687x16	5687x8
Downsampling	228x64	46x64	Upsampling	5687x8	28431x8
Fully Connected	46x64	8	Convolution	28431x8	28431x3

3.3.5 High resolution appearance model

As also shown in HTML [8], in order to train a texture model, the hand scans need to be brought into correspondence. To achieve this optimally, a graphics artist designed a UV hand template and used it as a reference template to unwrap the scans. However, the hand scans were acquired using constrained light conditions with baked shadows. As a result, before carrying out any training procedure, a pre-processing step need to be followed on the UV textures to remove the shading and illumination. In particular, PCA was applied to the UV textures in order to identify the components that mostly describe the shading factors. Then, those components were subtracted from each texture UV map to remove their unnecessary shading. Finally, an image processing step took place to map hand textures to more natural colors, which entailed increasing the brightness, gamma correction, and slightly adjusting the hue value.

For the training process, rather than modeling the appearance space in a low frequency PCA domain as other methods do [8], a powerful GAN architecture, namely StyleGAN

[123], was utilized to model the hand textures. Given the limited number of collected data, a smaller learning rate of 0.001 was used along with a regularization weight γ of 50 that further assisted in the "Fréchet Inception Distance" (FID) [124] score as well as the visual quality of the final results. In Figure 3.8, some random generations of the proposed high fidelity appearance model are illustrated. By utilizing the GAN architecture, high frequency skin details are preserved while avoiding the smoothness that may be introduced by the PCA model. Qualitative results of the proposed texture reconstruction in Figures 3.1 and 3.13 can validate this premise.

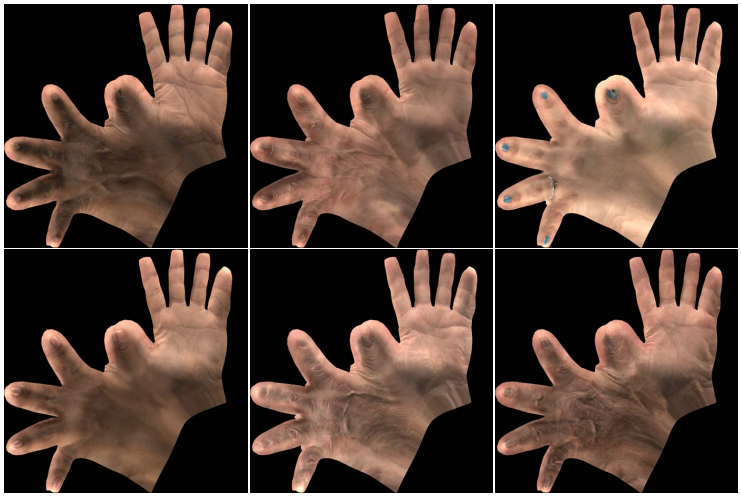


Figure 3.8: Generated high quality texture UV maps from the proposed GAN appearance model.

Interpolation on the latent space of the Texture Model: Apart from the high quality of the generated hand textures, a very handful property of the styleGAN is the smooth transitions of the latent space of the proposed texture model. To showcase the interpolation capabilities of the proposed style-based GAN model utilized to model the hand textures, an interpolation experiment was conducted. In particular, random pairs of UV maps were selected, projected to the latent space of the texture GAN and then interpolation was performed to their latent values. Figure 3.9 shows that the texture model produces meaningful latent representations between the two UV maps. Additionally, the generated UV maps are realistic having also very smooth transitions from the source to the target UV maps.

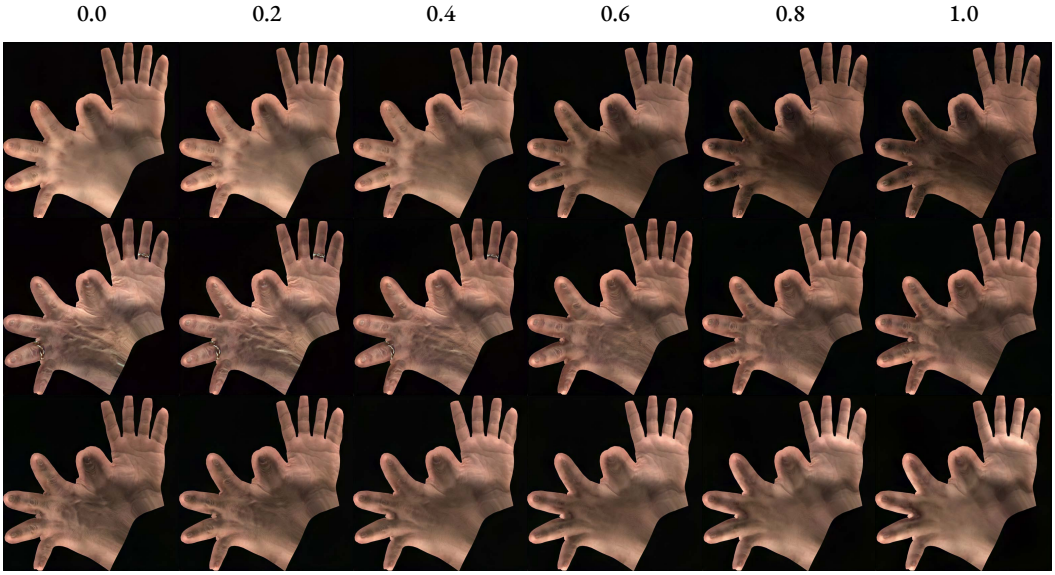


Figure 3.9: Interpolation on the latent space of the proposed texture model.

3.4 Experiments

3.4.1 Intrinsic Evaluation of Proposed Hand Model

In this Section, an initial evaluation is performed contrasting the proposed Handy++ hand shape model, against with the commonly used MANO model [2] and the PCA baseline Handy. Following common practice, comparison of the three models is performed in terms of *generalization* and *specificity*. For a fair comparison, the principal components of the PCA models are contrasted with the latent parameters of the "Handy++" model. In addition, to showcase the superiority of the proposed large scale dataset compared to the 30-subjects dataset used to train MANO, the *compactness* of the two PCA models is also reported. The three models were tested on the test split of the MANO dataset. To fairly compare the two PCA models, Handy and MANO, a variation of the Handy that utilizes the MANO template as described in Section 3.3.2, is also reported. Note that only the first 10 out of 31 principal components of MANO are publicly available.

Compactness. In Figure 3.11 , the compactness of the two models is reported. Compactness refers to the percentage of variance in the training dataset explained by the model for a given number of retained principal components. The figure shows that Handy model, trained on the proposed large scale dataset, better explains the variations in the dataset,

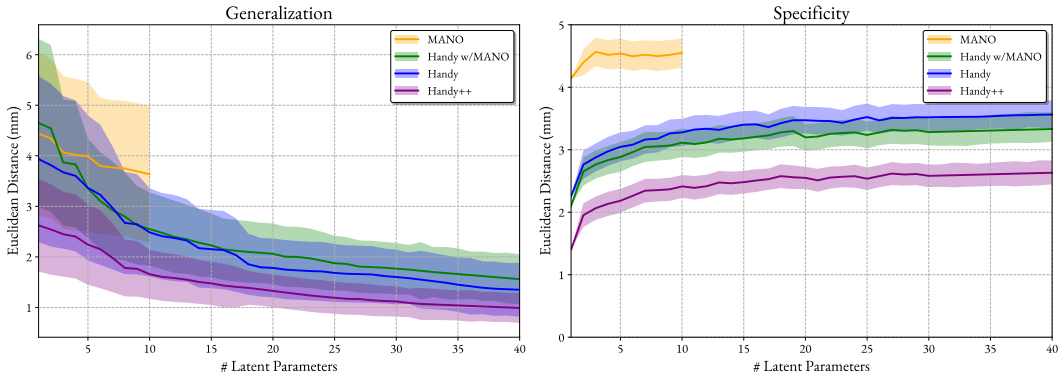


Figure 3.10: Evaluation of generalization and specificity against Handy and MANO models. The number of latent parameters refer to the number of principal components retained for the PCA models (Handy and MANO) or to the latent space size of Handy++.

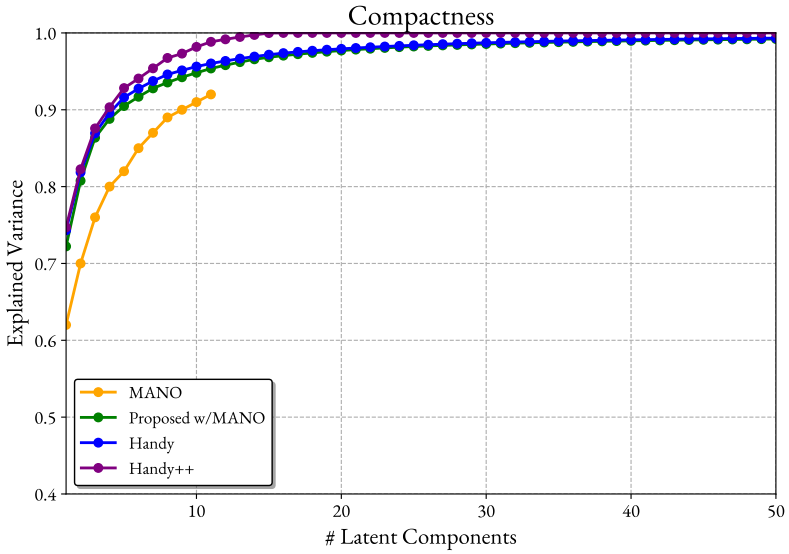


Figure 3.11: Evaluation of compactness between Handy and MANO models.

reaching the threshold of 90% variance from the 5th component, compared to the MANO model which reaches 90% variance at the 9th component.

Generalization demonstrates the ability of the model to generate new hand instances that were not present in the training set. To evaluate the models in terms of generalization, the MANO test set is utilized. In particular, the generalization error is measured as the mean per-vertex distance of each mesh on the MANO test set and its corresponding model reprojection. Figure 3.10 (left) reveals that the proposed "Handy++" model achieves better

out-of-distribution generalization and lower standard deviation compared to PCA models, MANO and Handy. More specifically, "Handy++" achieves a generalisation error lower than 2mm using a latent size lower than 10, compared to MANO that fails to generate novel hand shapes and has a 3.7mm error using 10 components. This highlights the argument that the mesh convolution autoencoder can attain more diverse samples using more compact representations.

Specificity. Finally, the specificity error is reported, which measures the realism of the generated hand shapes and their similarity to the training samples. The specificity error can be described as the distance of a generated sample from the model with its closest sample on the dataset. In practice, to measure the specificity error, 1,000 hand shapes were generated from each model and the per-vertex distance from the closest sample in the ground-truth datasets was measured. Similar to [125], given the small amount of training data we measure specificity on the training set. For a fair comparison, the samples used to train each model serve as ground-truth shapes. Figure 3.10 (right) shows that the proposed method Handy++ results into less specificity error compared to the MANO model by approximately 3.5mm. Furthermore, the proposed graph based autoencoder outperforms PCA-based Handy, by 1mm while at the same time it achieves lower deviation. Note that the slight deviation between the Handy and the Handy w/MANO models is attributed by the high resolution (8704 vertices) of the proposed hand template which leads to more detailed shapes compared to the MANO template (778 vertices).

3.4.2 Reconstruction of children's hands

A major limitation of current state-of-the-art hand models is that they were trained using limited data from specific age groups that do not reflect real hand variations. Given that the anatomy of children's hand is completely different compared to adult hand, current hand models fail to accurately reconstruct them. In this experiment, the case of reconstructing children's hand below the age of 12 was examined. Using 20 children hands that were not present in the training set, a fitting process was performed for each of the models. Table 3.2 highlights the reconstruction capabilities of the proposed hand model that was built with 1208 subjects with diverse age groups compared to the commonly used MANO model, which is composed of only 31 adult hands. Figure 3.12 shows the color-coded per vertex error which validates the superiority of the proposed model in children's hands reconstructions. As expected, MANO model fails to properly reconstruct the main anatomical difference between adults' and children's hands, which mostly lies on the back of the hand. Comparing the proposed Handy++ with Handy, it can be easily seen that Handy++ manages to attain almost half of the reconstruction error using the same

Table 3.2: Per vertex reconstruction error on 20 children's hands in mm (12 of them are less than 8 years old). We also report the performance of Handy with the MANO template (w/MANO) and use a different number of components (n_c) and latent codes (n_z) for a fair comparison. Bold denotes the best performance.

	Age <8	Age <12
MANO [2]	0.78	0.77
Handy w/ MANO (n_c = 10)	0.48	0.44
Handy w/ MANO (n_c = 30)	0.28	0.25
Handy (n_c = 10)	0.44	0.42
Handy (n_c = 30)	0.24	0.21
Handy++ (n_z = 8)	0.20	0.19
Handy++ (n_z = 30)	0.12	0.12

latent space size. Although more latent components could lead to even better results for "Handy++", as shown in Figure 3.10, a latent space of 8 components was used to showcase the superiority of "Handy++" using such a compact latent representation. Visually, as seen in Figure 3.12, this formulates to a small error, in almost all parts of the hand.

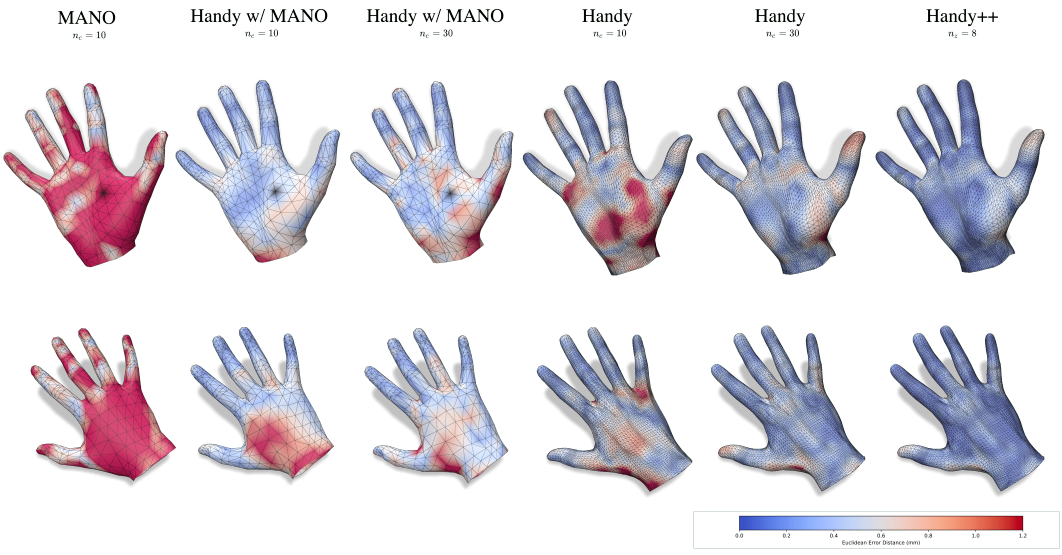


Figure 3.12: Color coded average reconstruction error of children's hands. The latent size of Handy++ is denoted by n_z whereas n_c corresponds to the components of the PCA models.

Figure 3.13: Hand shape and appearance reconstructions from single "in-the-wild" images. From left to right: i) "in-the-wild" image, ii) Handy-Shape reconstruction, iii) Handy-GAN result, iv) Handy-PCA model, and v) the HTML [8] texture on top of the shape mesh reconstructed using Handy++.

3.4.3 3D Reconstruction from single images

Following the pathway of many hand pose estimation methods, a synthetic dataset was created to train the hand reconstruction model. In particular, 30,000 texture images were generated from the GAN model to curate a synthetic dataset with textured hands. To increase the realism of the synthetic data, similar to [88], hands were rendered while interacting with objects of the ShapeNet dataset. The hands were also stitched to the SMPL body model using random shapes. In contrast to the Obman dataset [88], high resolution hand textures were used to bridge the domain gap between synthetic and "in-the-wild" hand images. To increase the diversity of the synthetic renderings, several illuminations, lighting, and camera configurations were used. In total, 90.000 synthetic images were created to trained the proposed regression model. Samples of the generated synthetic dataset are illustrated in Figure 3.14.

Figure 3.14: Samples of the synthetic dataset.

In order to leverage the latent space of the proposed Handy++ model, an off-the-shelf method [87, 88, 126] was modified by substituting the MANO parametric model with Handy++. For comparison reasons, Handy and MANO models were also used as a shape decoders. Unlike previous methods that neglect the texture reconstruction, two extra branches were added to regress the latent space $\mathbf{w}$ of the texture model and the camera configuration $(s, \mathbf{t})$. The network was trained using a set of loss functions that enable

accurate hand pose, shape, and appearance estimation. In particular, similar to [87, 88, 1], a loss was applied on both the latent shape and pose parameters and the generated 3D vertex positions to enforce shape and pose estimation:

$$\begin{gathered}\mathcal{L}_{\beta} = \|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|_2, \quad \mathcal{L}_{\theta} = \|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\|_2 \\ \mathcal{L}_{3D} = \sum_i \|\mathbf{v}_i - \hat{\mathbf{v}}_i\|_2\end{gathered} \tag{3.6}$$

where $\boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{v}$ denote the predicted shape, pose and vertex positions, respectively, and $\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\theta}}, \hat{\mathbf{v}}$ their corresponding ground-truth values.

To precisely generate hand textures, a combination of loss functions was used. Given that the synthetic data were rendered using known ground-truth UV maps, the model was directly enforced to produce textures that match the ground-truth UV maps with a UV loss:

$$\mathcal{L}_{uv} = \|UV^{\mathcal{G}} - UV^{0}\|_1 \tag{3.7}$$

where $UV^{\mathcal{G}}$ corresponds to the generated UV texture and UV^{O} to the ground-truth texture.

Additionally, a differentiable renderer was used using an orthographic camera with trainable parameters that projects the generated 3D hand on the input image plane. A pixel loss between the rendered image and the input image was utilized in order to obtain accurate camera parameters and model the details of the appearance:

$$\mathcal{L}_{pix} = \|I^{\mathcal{R}} - I^{0}\|_1 \tag{3.8}$$

with $I^{\mathcal{R}}, I^{0}$ the original and the rendered images, respectively.

Finally, to constrain the generated hand textures, a perceptual loss [127] was applied that imposes the texture model to produce realistic textures that match the input image:

$$\mathcal{L}_{lpips} = \mathcal{F}(I^{\mathcal{R}}, I^{0}) \tag{3.9}$$

The overall loss can be then defined as follows:

$$\mathcal{L} = \mathcal{L}_{3D} + 0.1\mathcal{L}_{uv} + 0.1\mathcal{L}_{pix} + 0.01\mathcal{L}_{lpips} + 10\mathcal{L}_{\beta} + 10\mathcal{L}_{\theta} \tag{3.10}$$

Although synthetic data can be sufficient to train a hand pose and appearance estimation network, they usually constrain the texture regressor to latent codes that lie within the distribution of the textures, failing to reconstruct more challenging textures. In order to boost high fidelity appearance reconstruction, a set of "in-the-wild" images was collected

and their corresponding Handy pose, shape and texture parameters were predicted using the pre-trained regression network. Then, similar to [128], only the texture parameters $\mathbf{w}$ were further optimized in order to generate high resolution textures that match the appearance of the "in-the-wild" images. The optimization function is constructed with Eq. 3.8, 3.10, along with a L_2 regularization on $\mathbf{w}$ to secure that it does not greatly deviate from the initial estimation. Once the improved $\mathbf{w}'$ were acquired, a fine-tuning process of the regression network on the "in-the-wild" dataset was performed.

Implementation Details:
The 3D reconstruction method from single images is composed by three main components. The first module is a ResNet50 network, pretrained on ImageNet, that acts as a feature extractor. Following that, a set of regression branches that predict the latent parameters of the Handy model. In the case of Handy this translates to shape, pose and texture parameters whereas in the case of Handy++ model, the regression branches regresses only the decoder latent space along with the texture parameters. Finally, the last module of the proposed method predicts the parameters (scale and translation) of an orthographic camera that is used to render the predicted hand mesh back to the image space. All of the aforementioned branches are composed by an MLP layer and take as input the latent features of ResNet. The full architecture is depicted in Figure 3.15. The proposed architecture is trained for 250 epochs with the Adam optimizer and a learning rate of 5e-5.

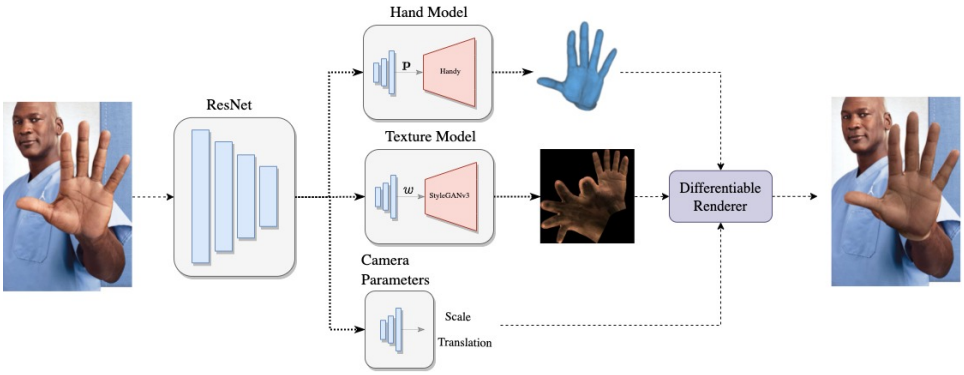


Figure 3.15: Architecture of the proposed 3d hand reconstruction method. The ResNet latent features are processed by three parallel regression methods, i.e. hand shape/pose regressor with parameters $\mathbf{z}$, texture regressor with parameters $\mathbf{w}$ and the camera regressor predicting camera parameters.

Reconstruction under controlled Conditions:
To quantitatively assess the texture reconstruction of the proposed method, images from the scanning device were utilized. In such case, the corresponding UV maps of each subject acquired after the registration step can act as ground-truth UV textures. As it can be

Table 3.3: Quantitative comparison between the texture reconstruction models under controlled conditions.

Method	L_1 ($\times 10^{-2}$)	LPIPS [127]
HTML	2.14	0.092
Handy-PCA	1.44	0.065
Handy-GAN	**0.47**	**0.010**

observed from the figures reported in Table 3.3, the proposed texture model outperforms the HTML model by a significant margin, in terms of L_1 and LPIPS losses. The superiority of the proposed method can be also validated in Figure 3.16. To properly compare the texture reconstruction of each method, all three methods share the same shape and pose extracted from the proposed regression network. The proposed method can reconstruct high frequency details of the input image such as wrinkles, rings, tattoos, and nail polish. In contrast, PCA-based methods produce smooth results that lack high frequency details and even fail to properly reconstruct the skin color (Figure 3.16, row 2).

Reconstruction from "in-the-wild" images:
Furthermore, the proposed method was qualitatively compared against the HTML method in an unconstrained setting using "in-the-wild" images. In Figure 3.13, a comparison between the three methods is depicted, using challenging figures with different skin colors, shape structures, and light conditions. Similar to the previous experiment, all three methods share the same shape and pose. As can be easily observed, Handy-GAN can reconstruct high frequency details such as wrinkles and precise hand colors, even with hands that are out of the trained distribution. It is also important to note that Handy-GAN can also reconstruct textures from hands with vitiligo disorder that have severe color discontinuities.

Finally, to quantitatively evaluate shape and pose reconstruction under "in-the-wild" conditions, the proposed model was compared with several state-of-the-art models along with the PCA alternatives on the popular benchmark dataset FreiHand [1]. Table 3.4 shows that the proposed method outperforms current state-of-the-art model-based methods utilizing MANO as their backbone. It is also important to note that, as expected, the proposed method trained on the proposed synthetic dataset (w/Synthetic), achieves better hand reconstructions compared to the method trained with the Obman dataset [88] (w/Obman). This finding highlights and validates the assumptions that the proposed synthetic dataset bridges the domain gap between synthetic and "in-the-wild" images. Finally, it is worth mentioning that although the PCA-based Handy achieves remarkable results compared to MANO model, it can not compete the neural mesh autoencoder Handy++ which is able to achieve even 1.4mm less reconstruction error. Qualitative results of the proposed hand

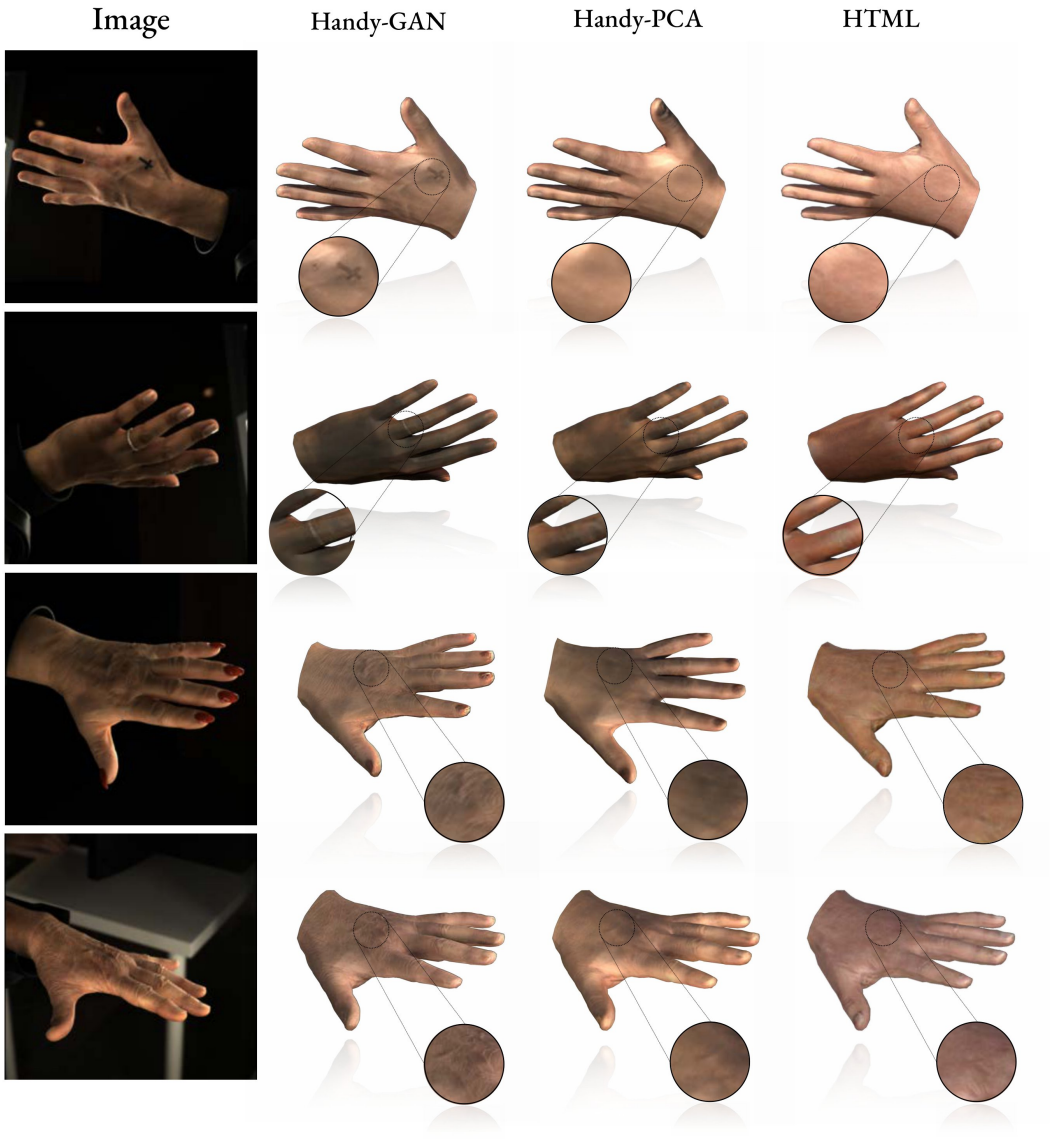


Figure 3.16: Hand shape and appearance reconstructions from single images under controlled conditions.

reconstruction are shown in Figure 3.17.

Method	MPVPE ↓	MPJPE ↓	F@5 mm ↑	F@15 mm ↑
Hasson *et.al.*[88]	13.2	-	0.436	0.908
Boukhayma *et.al.*[87]	13.	-	0.435	0.898
MANO CNN [1]	10.8		0.529	0.935
MANO FIT [1]	13.7	-	0.439	0.892
HTML [8]	11.1	11.0	0.508	0.930
S^2Hand [129]	11.8	11.9	0.481	0.920
Ren *et.al.*[130]	8.1	8.0	0.649	0.966
Handy w/Obman	9.9	9.7	0.572	0.922
Handy++ w/Obman	8.5	8.6	0.624	0.946
Handy w/Synthetic	8.8	8.7	0.612	0.952
Handy++ w/Synthetic	7.8	7.7	0.662	0.964
Handy	7.8	7.8	0.654	0.971
Handy++	**6.9**	**7.1**	**0.701**	**0.987**

Table 3.4: Quantitative comparison on the FreiHand dataset [1]. We evaluate the proposed and the baseline methods in terms of mean per joint position error (MPJPE), mean per vertex position error (MPVPE). Additionally, we report F-score at a given threshold d (F@d) which is the harmonic mean of precision and recall.

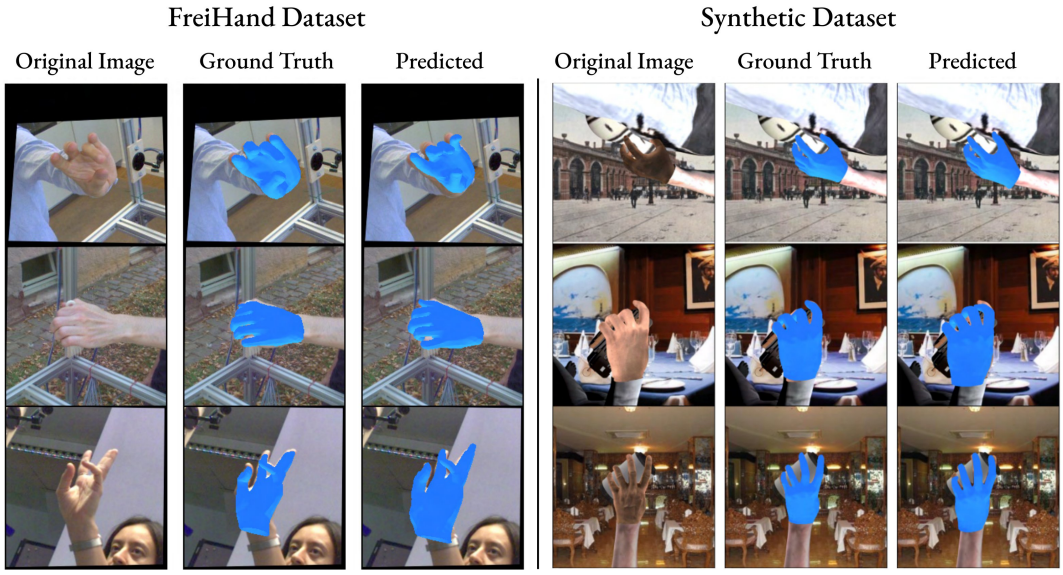


Figure 3.17: Shape and pose reconstructions from the FreiHand [1] and the proposed synthetic dataset.

3.4.4 Reconstruction from Point Clouds

Apart from the reconstruction from single images, the proposed method was also evaluated on shape and pose reconstructions from point clouds. In particular, the proposed Handy++ model was compared with the state-of-the-art implicit hand model LISA [85]

on the registered MANO dataset [2]. Following [85], $100K$ points were sampled from the surface of the MANO scans and a fitting optimization was performed using the Chamfer distance between the sampled points and Handy template. To evaluate the fitting, the vertex-to-point distances (in mm) from the reconstruction to the scan (R2S) point cloud and the other way around (S2R) were measured. Table 3.5 shows that the proposed model achieves a lower reconstruction error with only 8 latent codes, which translates to less than 10 shape components, outperforming LISA, MANO and Handy models.

Table 3.5: Reconstruction error on point clouds sampled from the MANO dataset [2].

	R2S [mm]	S2R [mm]
MANO [2]	2.90	1.52
LISA-im [85]	1.96	1.13
LISA [85]	0.64	0.58
Handy w/MANO ($n_c = 10$)	0.21	0.29
Handy w/MANO ($n_c = 30$)	0.12	0.21
Handy ($n_c = 10$)	0.16	0.25
Handy ($n_c = 30$)	0.11	0.19
Handy++ ($n_z = 8$)	**0.08**	**0.12**

3.5 Conclusion

In this chapter, the first large-scale shape and appearance hand model, named *Handy++*, was introduced and presented. The proposed model was trained with over 1200 subjects with large demographic diversity, overcoming the limitations of previous parametric models to reconstruct hands from diverse distributions, such as the shape of children's hands. The proposed model is structured with an encoder-decoder architecture that utilizes spiral mesh convolutions. In contrast to traditional linear PCA models, the graph-based model gains more expressive power and manages to outperform PCA models while having more compact representations. Additionally, a style-based GAN was trained to generate UV textures with high frequency details that traditional PCA methods fail to model. Extending comparison experiments showcase that Handy++ achieves remarkable results highlighting and demonstrating its expressive power to reconstruct challenging hand shapes and appearances.

CHAPTER 4

DYNAMIC DEFORMABLE MODELS

Contents

The recent advances in deep learning have significantly pushed the state-of-the-art in photorealistic video animation given a single image. In this chapter, those advances are extrapolated to the 3D domain, by studying 3D image-to-video translation with a particular focus on dynamic 3D facial expressions. Although 3D facial generative models have been widely explored during the past years, 3D animation remains relatively unexplored. To this end, a deep mesh encoder-decoder like architecture was employed to synthesize realistic high resolution facial expressions by using a single neutral frame along with an expression identification. In addition, processing 3D meshes remains a non-trivial task compared to data that live on grid-like structures, such as images. Given the recent progress in mesh processing with graph convolutions, a recently introduced learnable operator was utilized, which acts directly on the mesh structure by taking advantage of local vertex orderings. In order to generalize to 4D facial expressions across subjects, the proposed model was trained using a high resolution dataset with 4D scans of six facial expressions from 180 subjects [131]. Experimental results demonstrate that the proposed approach preserves the subject's identity characteristics even for unseen subjects and generates high quality expressions.

4.1 Introduction

Recently, facial animation has received attention from the industrial graphics, gaming and filming communities. Face is capable to impart a wide range of information not only about the subject's emotional state but also about the tension of the moment in general. An engaged and crucial task is 3D avatar animation, that has lately become feasible [132]. With modern technology, a 3D avatar can be generated by a single uncalibrated camera [133] or even by a self portrait image [134]. At the same time, capturing facial expression is an important task in order to perceive behaviour and emotions of people. To tackle the problem of facial expression generation, it is essential to understand and model facial muscle activations that are related to various emotions. Several studies have attempted to decompose facial expressions on two dimensional spaces such as images and videos [135, 136, 137, 138]. However, modeling facial expressions on high resolution 3D meshes remains unexplored.

In contrast, few studies have attempted 3D speech-driven facial animation exclusively based on vocal audio and identity information [139, 140]. Nevertheless, emotional reactions of a subject are not always expressed vocally and speech-driven facial animation approaches neglect the importance of facial expressions. For instance, sadness and happiness are two very common emotions that can be voiced, mainly, through facial deformation. To this end, facial expressions are a major component of entertainment industry, and can convey emotional state of both scene and identity.

People signify their emotions using facial expressions in similar manners. For instance, people express their happiness by mouth and cheek deformations, that vary according to the subject's emotional state and characteristics. Thus, one can describe expressions as "unimodal" distributions [137], with gradual changes from the neutral model till the apex state. Similarly to speech signals, emotion expressions are highly correlated to facial motion, but lie in two different domains. Modeling the relation between these two domains is essential for the task of realistic facial animation. However, in order to disentangle identity information and facial expression it is essential to have a sufficient amount of data. Although most of the publicly available 3D datasets contain a large variety of facial expression, they are captured only from a few subjects. Due to this difficulty, prior work has only focused on generating expressions in 2D.

In this chapter a concrete methodology is presented for the generation of 3D facial animation given a target expression and a static neutral face. Synthesis of facial expression generation on new subjects can be achieved by expression transfer of generalized deformations [141, 136]. In order to produce realistic expressions, the modeling of the facial

animation is done directly on the mesh space, avoiding to focus on specific face landmarks. Specifically, the proposed method comprises two parts: (a) a recurrent LSTM encoder to project the expected expression motion to an expression latent space, and (b) a mesh decoder to decode each latent time-sample to a mesh deformation, which is added to the neutral expression identity mesh. The mesh decoder utilizes intrinsic lightweight mesh convolutions, introduced in [7], along with unpooling operations that act directly on the mesh space [122]. The model was trained in an end-to-end fashion on a large scale 4D face dataset.

To summarize the contributions of the work presented in this chapter are:

- A methodology that tackles a novel and unexplored problem, i.e. the generation of 4D expressions given a single neutral expression mesh.
- A method that considerably deviates from methods in the literature as it can be used to generate 4D full-face customised expressions on real-time.
- A fully customisable method where both the desired length and the target expression can be controlled by the user.
- The first 3D facial animation framework that utilizes an intrinsic encoder-decoder architecture that operates directly on mesh space using mesh convolutions instead of fully connected layers, as opposed to state-of-the-art methods like [140, 139].

4.2 Related Work

Facial animation generation

Following the progress of 3D Morphable Models (3DMMs), several approaches have attempted to decouple expression and identity subspaces and built linear [142, 143, 144] and nonlinear [145, 122, 7] expression morphable models. However, all of the aforementioned studies are focused on static 3D meshes and they cannot model 3D facial motion. Recently, a few studies attempted to model the relation between speech and facial deformation for the task of 3D facial motion synthesis. Although 3D generation and 3DMMs are widely explored, there are just a few studies that focus on synthesis of 3D facial motion. In particular, most of these studies focus on the relation between speech and facial deformations. Karras et al. [139] modeled speech formant relationships with 5K vertex positions, generating facial motion from LPC audio features. While this was the first approach to

tackle facial motion directly on 3D meshes, their model is subject specific and cannot be generalized across different subjects. Towards the same direction, in [140], facial animation was generated using a static neutral template of the identity and a speech signal, used along with DeepSpeech [146] to generate more robust speech features. A different approach was utilized in [147], where 3D facial motion is generated by regressing on a set of action units, given MFCC audio features processed by Recurrent Neural Network (RNN) units. However, their model is trained on parameters extracted from 2D videos instead of 3D scans. Similarly, Pham et al. [148] used spatio-temporal convolutions to synthesize facial animation from speech using a blendshape model driven by action unit intensities. Tzirakis et al. [149] combined predicted blendshape coefficients with a mean face to synthesize 3D facial motion from speech, replacing also fully connected layers, utilized in previous studies, with an LSTM. Blendshape coefficients are also predicted from audio, using attentive LSTMs in [150]. In contrast with the aforementioned studies, the proposed method aims to model facial animations directly on 3D meshes. Furthermore, although blendshape coefficients might be easily modeled, they rely on predefined face rigs, a factor that limits their generalization to new unseen subjects. Recently, Otberdout et al. [151] proposed a similar method that is able to regress dense 3D faces from sparse landmarks

Facial Expression datasets

Another major reason that 4D generative models have not been widely exploited is due to the limited amount of 3D datasets. During the past decade, several 3D face databases have been published. However, most of them are static [152, 153, 154, 155, 156, 157, 158], consisted of few subjects [122, 159, 160, 161], and have limited [162, 163, 153] or spontaneous expressions [164, 165], making them inappropriate for tasks such as facial expression synthesis. On the other hand, the recently proposed 4DFAB dataset [131] consists of six 3D dynamic facial expressions (from 180 subjects), which is ideal for subject independent facial expression generation. In contrast with all previously mentioned datasets, 4DFAB, due to the high range of subjects, presents and offers a promising resource towards disentangling facial expression from the identity information.

4.3 Method

The overall architecture of the proposed model is structured by two major components (see Figure 4.1). The first component implements a temporal encoder, using an LSTM

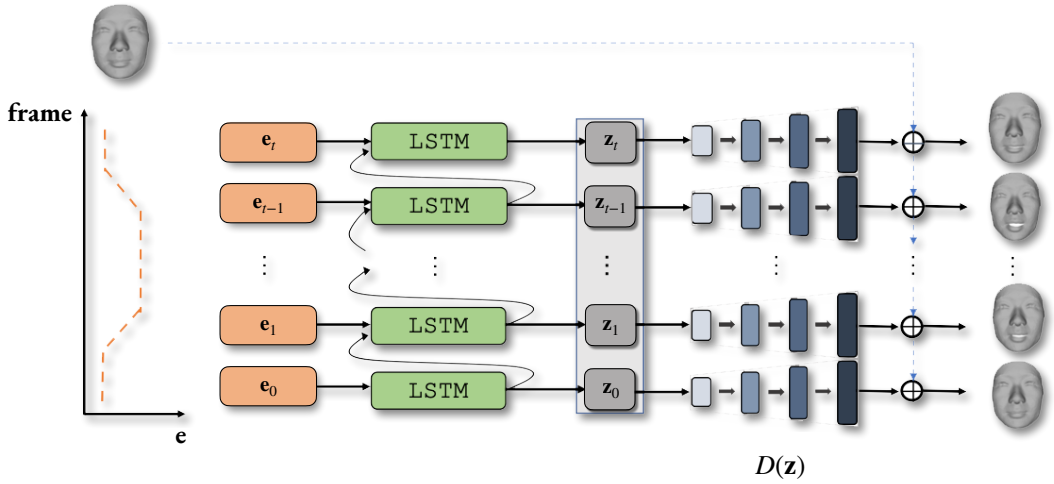


Figure 4.1: Network architecture of the proposed method.

layer that encodes the expected facial motion of the target expression. It takes as input a temporal signal $e \in R^{6\times T}$ with length T, equal to the target facial expression, also equipped with information about the time-stamps that show when the generated facial expression reaches onset, apex and offset modes. Each time-frame of signal e can be characterised as a one-hot encoding of one of the six expressions, with amplitude that indicates the scale of the expression. The second component of the proposed network consists of a frame decoder, with four layers of Spiral mesh convolutions [7], where each one is followed by an upsampling layer. Each upsampling layer increases the number of vertices by five times, and every mesh convolution is followed by a ReLU activation [166]. Finally, the output of the decoder is added to the identity neutral face. Given a time sample from the latent space, the frame decoder network models the expected deformations on the neutral face. Each output time frame can be expressed as:

$$\begin{aligned} \hat{x}_t &= D(z_t) + x_{id}, \\ z_t &= E(e_t) \end{aligned} \tag{4.1}$$

where $D(\cdot)$ denotes the mesh decoder network, $E(\cdot)$ the LSTM encoder, e_t the facial motion information for time-frame t and x_{id} the neutral face of the identity. The network details can be found in Table 4.1. The proposed model was trained for 100 epochs with learning rate of 0.001 and a weight decay of 0.99 on every epoch. Adam optimizer [167] was used with a 5e-5 weight decay.

Loss function. The mesh decoder network outputs motion deformation for each time-

frame with respect to the expected facial animation. To train the proposed model both the reconstruction error L_r and the temporal coherence L_c were optimized, as proposed in [139]. Specifically, the loss function between the generated time frame $\hat{x}_t$ and its ground truth x_t value is defined as:

$$\begin{aligned} L_r(\hat{x}_t, x_t) &= \|\hat{x}_t - x_t\|_1 \\ L_c(\hat{x}_t, x_t) &= \|(\hat{x}_t - \hat{x}_{t-1}) - (x_t - x_{t-1})\|_1 \\ L(\hat{x}_t, x_t) &= L_r(\hat{x}_t, x_t) + L_c(\hat{x}_t, x_t) \end{aligned} \tag{4.2}$$

Although reconstruction loss term L_r can be sufficient to encourage model to match ground truth vertices at each time step, it may not produce high-quality realistic animation. On the contrary, temporal coherence loss L_c term ensures temporal stability of the generated frames by matching the distances between consecutive ground truth frames and generated expressions.

Table 4.1: Mesh Decoder architecture

Layer	Input Dimension	Output Dimension
Fully Connected	64	46x64
Upsampling	46x64	228x64
Convolution	228x64	228x32
Upsampling	228x32	1138x32
Convolution	1138x32	1138x16
Upsampling	1138x16	5687x16
Convolution	5687x16	5687x8
Upsampling	5687x8	28431x8
Convolution	28431x8	28431x3

4.4 Experiments

4.4.1 Dynamic 3D face database

To train the proposed expression generative model the recently published 4DFAB [131] was used. 4DFAB contains dynamic 3D meshes of 180 people (60 females, 120 males) with ages between 5 to 75 years. The devised meshes display a variety of complex and exaggerated facial expressions, namely *happy, sad, surprise, angry, disgust* and *fear*. The 4DFAB database displays high variance in terms of ethnicity origins, including subjects from more than 30 different ethnic groups. The dataset was split into 153 subjects for training and 27 for testing. The data were captured with 60fps, thus each expression is

sampled every approximately 5 frames in order to allow the proposed model to generate extreme facial deformations. Given the high quality of the data (each mesh is composed by 28K vertices) as well as the relatively big number of subjects, 4DFAB presents a rich and rather challenging choice for training generative models.

Expression Motion Labels

In this study, the assumption that each expression can be characterised by four phases of its evolution (see Figure 4.2) is used. First, the subject starts from a neutral pose and at a certain point their face starts to deform, in order to express their emotional state. This phase is called the *onset phase*. After the subject's expression reaches its *apex state*, it will start again its deformation from the peak emotional state until it resides to its neutral state again. This phase is called *offset phase*. Thus, each time frame is assigned a label that reflects its emotional state phase. The emotional state is considered as a value ranging from 0 to 1 assigned to each frame, with 0 representing the neutral phase and 1 the apex phase. Onset and offset phases are represented via a linear interpolation between the apex and neutral phases (see Figure 4.2). However, expressions may also range in terms of extremeness, i.e. the level of intensity in subject's expression. To let the proposed model learn diverse extremeness levels for each expression, it is essential to scale each expression motion label from $[0, 1]$ to $[0, s_i]$, where $s_i \in (0, 1]$ represents the scaled value of the apex state according to the intensity of the expression. Intuitively, the extremeness of each expression is proportional to the absolute mean deformation of the expression, thus scaling factor s_i can be calculated as:

$$s_i = \frac{clip(\frac{m_i - \mu_e}{\sigma_e}) + 1}{2} \tag{4.3}$$

where m_i is a scalar value representing the absolute value of the mean deformation of the sequence from neutral frame and μ_e, σ_e the mean and standart deviation of the deformation of the respective expression. Clip() function is used to clip values to [-1,1].

4.4.2 Dynamic Facial Expressions

The proposed model for the generation of facial expressions is assessed both qualitatively and quantitatively. The model is trained by feeding the neutral frame of each subject and the manifested motion (i.e. the time-frames where the expression reaches onset, apex and offset modes) of the target expression. The performance of the proposed model was evaluated by its ability to generate expressions of 27 unobserved test subjects. To this

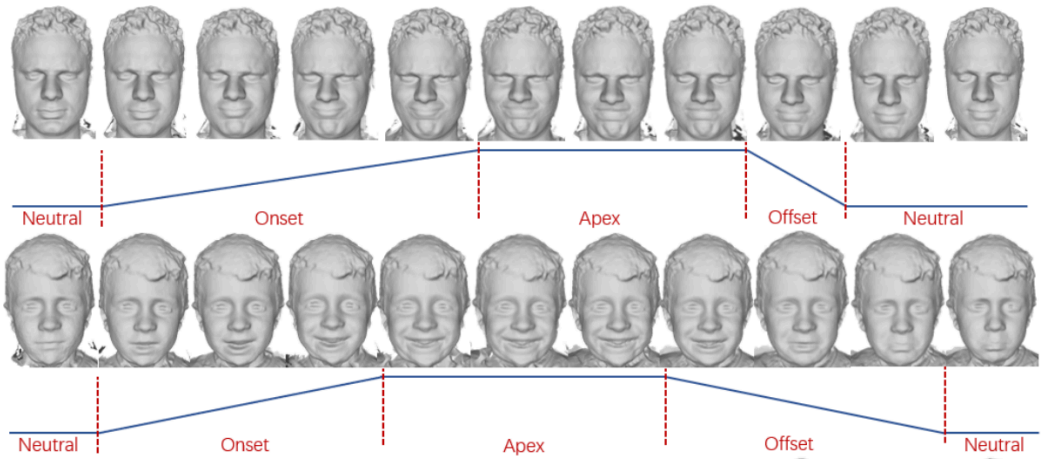


Figure 4.2: Sample subjects from the 4DFAB database posing an expression along with expression motion labels.

end, the reconstruction loss as the per-vertex Euclidean distance between each generated sample and its corresponding ground truth was calculated.

Baseline. As a comparison baseline an expression blendshape decoder was used, that transforms the latent representation z_t of each time-frame, i.e. the LSTM outputs, to an output mesh. In particular, expression blendshapes were modeled by first subtracting the neutral face of each subject to its corresponding expression, for all the corresponding video frames. With this operation its is possible to capture and model just the motion deformation of each expression. Then, Principal Component Analysis (PCA) was applied to the motion deformations to reduce each expression to a latent vector. For a fair comparison, the same latent size was used for both the baseline and the proposed method.

The results presented in Table 4.2 show that the proposed model outperforms the baseline with regards to all the expressions, as well as on the entire dataset (0.39mm vs 0.44mm).

Table 4.2: Generalization per-vertex loss over all expressions, along with the total loss.

Model	*Happy*	*Angry*	*Sad*	*Surprise*	*Fear*	*Disgust*	Total
PCA - Baseline	0.49	0.37	0.37	0.48	0.43	0.45	0.44
Proposed	**0.37**	**0.35**	**0.36**	**0.43**	**0.42**	**0.42**	**0.39**

Moreover, as observed by inspecting in Figure 4.3, the proposed method can produce more realistic animations, especially in the mouth region, compared to PCA blendshapes. Error visualizations in Figure 4.3, showcase that the blendshape model produces mild transitions between each frame and cannot generate extreme deformations. Note also that the errors

of the proposed method are mostly centered around the mouth and the eyebrows, due to the fact that the proposed model is subject independent and each identity expresses its emotions in different ways and varying extents. In other words, the proposed method models each expression with respect to the motion labels without taking into account identity information, thus the generated expressions can have some variations compared to the ground truth subject-dependent expression.

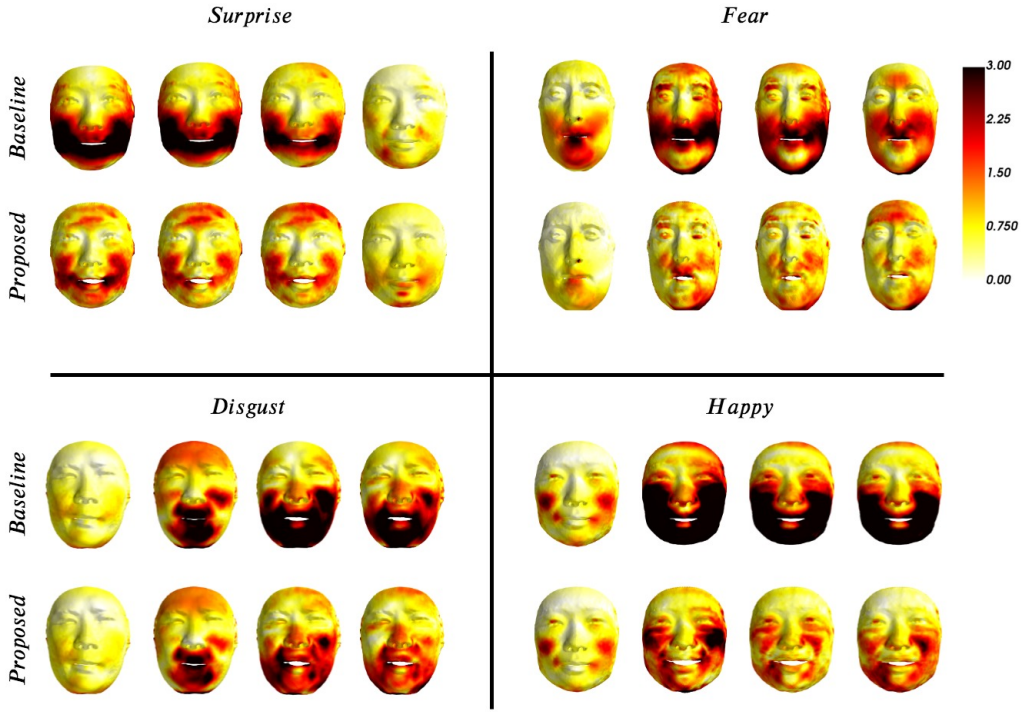


Figure 4.3: Color heatmap visualization of error metric of both baseline (top rows) and proposed (bottom rows) model against the ground truth test data for four different expressions.

For a subjective qualitative assessment, Figure 4.4 shows several expressions that were generated by the proposed method.

4.4.3 Classification of generated 4D expressions

To further assess the quality of the generated expressions a classifier was trained to identify expressions. The architecture of the classifier is based on a projection on the PCA space

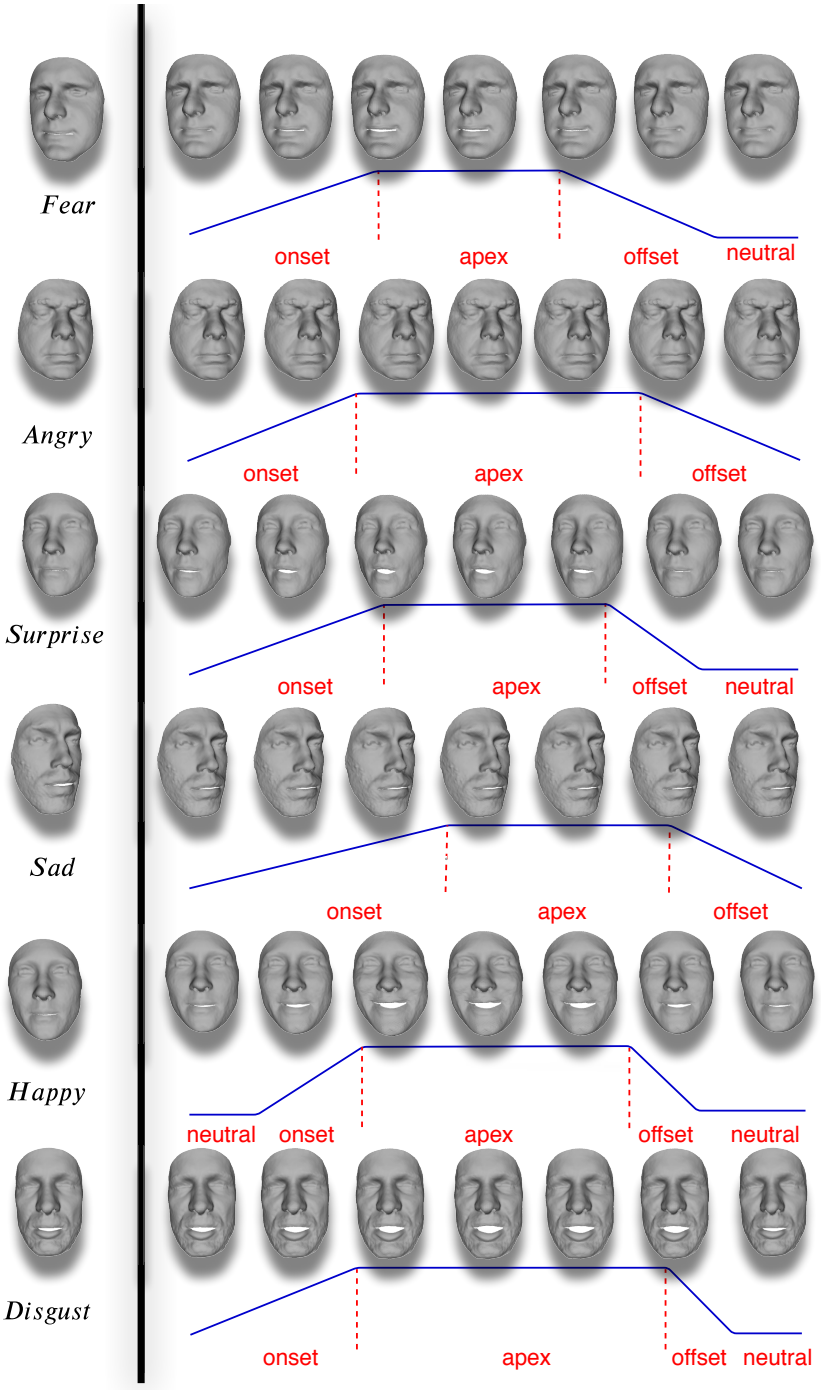


Figure 4.4: Frames of generated expressions along with their expected motion labels: Fear, Angry, Surprise, Sad, Happy, Disgust (from top to bottom).

followed by three fully connected layers. The sequence classification is performed in two steps. In particular, first 64-PCA coefficients were computed to represent all expression and deformation variations of the training set and then a frame encoder was used to map the unseen test data to the latent space. Following that, the latent representations of each frame were concatenated and processed by fully-connected layers in order to predict the expression of the given sequence. The network was trained on the same training set that was originally used for the generation of expressions, with Adam optimizer and 5e-3 weight decay for 13 epochs. Table 4.3 presents the achieved classification performance for the ground truth test data and the generated data from the proposed model and from the baseline model, respectively.

Table 4.3: Constructing classification performance between ground truth (test) data, generated (by the proposed method) data, and the blendshape baseline.

	PCA - Baseline			**Proposed**			Ground Truth		
	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
Surprise	0.69	0.62	0.65	**0.74**	**0.90**	**0.81**	0.69	0.83	0.75
Angry	0.67	0.55	0.60	0.75	0.52	**0.61**	**0.81**	**0.59**	0.60
Disgust	0.51	0.72	0.60	**0.65**	**0.83**	**0.73**	0.58	0.72	0.65
Fear	0.59	0.36	0.44	**0.67**	**0.43**	**0.52**	0.64	0.32	0.43
Happy	0.63	0.59	0.60	0.71	0.86	0.78	**0.73**	**0.93**	**0.82**
Sad	0.43	0.57	0.49	0.62	0.60	0.61	**0.74**	**0.77**	**0.75**
Total	0.58	0.57	0.57	0.69	0.69	0.68	**0.70**	**0.70**	**0.68**

As can be observed from the figures in Table 4.3, the generated data from the proposed model achieve similar classification performance with ground truth data across almost every expression. In particular, the generated *surprise, disgust* and *fear* expressions generated by the proposed model can be even easier classified compared to the ground truth test data. Note also that both ground truth data and the data generated by the proposed model achieve 0.68 F1-score in total.

4.4.4 Loss per frame

Since the overall loss is calculated for all frames of the generated expression, it is not possible to assess the ability of the proposed model to generate the onset, apex, and offset of each expression with low error-rate. The performance of the model on each expression phase was evaluated by calculating the average L_1 distance between the generated and corresponding ground truth frame for each frame of the evolved expression. Figure 4.5 illustrates that the apex phase, typically occurring between time-frames 30-80, exhibits an increased L_1 error for both models. However, the proposed method demonstrates a

stable loss of approximately 0.45mm throughout the apex phase, whereas the blendshape baseline struggles to model the extreme deformations characteristic of the apex phase of the expression.

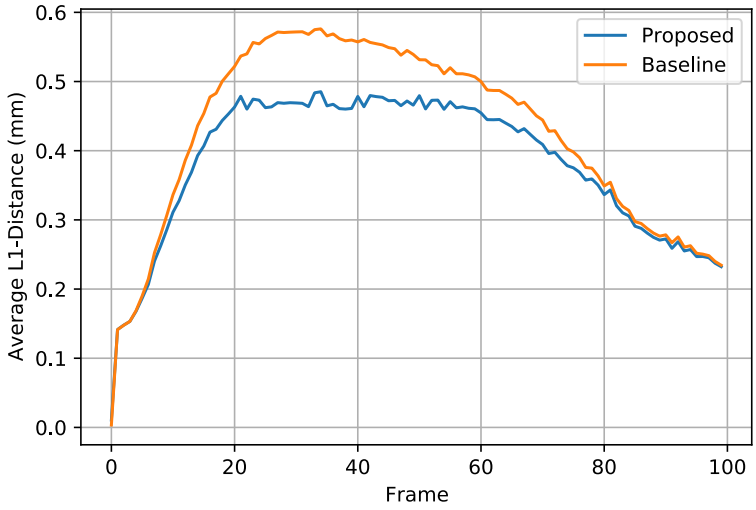


Figure 4.5: Average per-frame L_1 error between the proposed method and the PCA-based blendshape baseline.

4.4.5 Interpolation on the latent space

To qualitatively evaluate the representation power of the proposed LSTM encoder, linear interpolation was applied to the expression latent space. Specifically, two different apex expression labels were chosen from the test set and encoded using the LSTM encoder to two latent variables z_0 and z_1, each one of size 64. All intermediate encodings were then produced by linearly interpolating the line between them, i.e., $z_\alpha = \alpha z_1 + (1-\alpha)z_0$, where $\alpha \in (0,1)$. The latent samples z_α were subsequently fed to the mesh decoder network. The interpolations between different expressions are visualized in Figure 4.6.

4.4.6 Expression generation in-the-wild

Given the significance of expression generation in graphics and film industries, a real-world application of the 3D facial expression generator is proposed. In particular, several image pairs with neutral and various expressions of the same identity were collected for

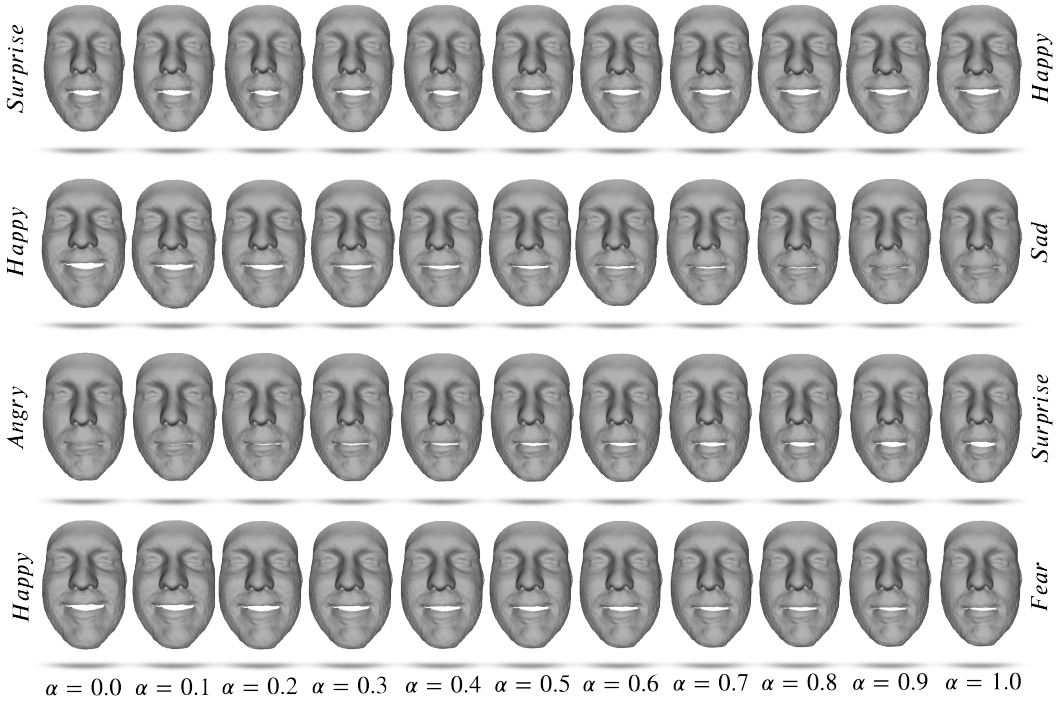


Figure 4.6: Interpolation on the latent space between different expressions.

the purpose of realistically synthesizing the 4D animation of the target expression. To acquire a neutral 3D template for animation purposes, a fitting methodology was applied to the neutral image as proposed in [9]. Utilizing the fitted neutral mesh, the proposed model was employed to generate several target expressions, as shown in Figure 4.7. The proposed method demonstrates the ability to synthesize a series of realistic 3D facial expressions, showcasing the framework's capability to animate a template mesh with a desired expression. To qualitative evaluate the similarity of the generated expression with the target expression of the same identity, GANFit [9] was fitted on a different image of the identity posing the desired expression.

4.5 Limitations and Future Work

Although the proposed framework can model and generate a wide range of realistic expressions, it cannot thoroughly model extreme variations. As mentioned in section 4.4.1, an attempt was made to adapt the extremeness variations of each subject into the training procedure using an intuitive scaling trick. However, it is not certain that the mean absolute deformation of each mesh always accurately represents the extremeness of the

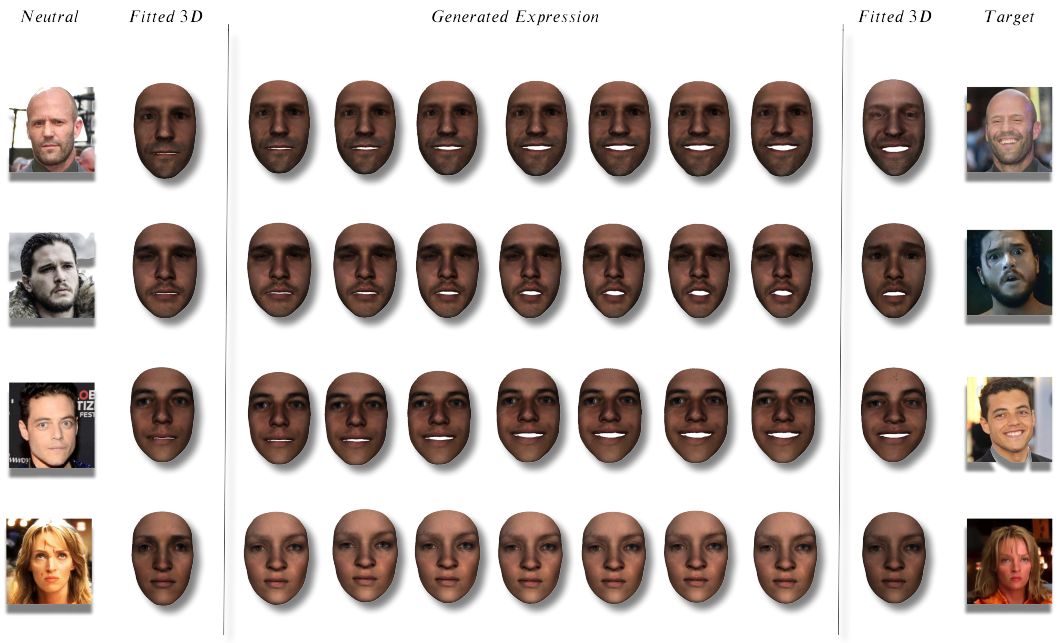


Figure 4.7: Generation of expressions in-the-wild form 2D images. Both left and right fitted 3D faces are obtained using GANFit [9].

conducted expression. Furthermore, due to the limited number of subjects, a generic model was built. Using this formulation, the model learns to generate displacements given a certain expression label, which results in deterministic facial expressions. To overcome this limitation, future work will involve extending the training procedure by including more datasets and increasing the number of subjects to create a person-specific expression model. Using a subject specific model, one can generate expressions conditioned on the identity, which can better approximate realistic expressions. Thus, one of the future tasks is to adjust the model in order to generate fully-customized realistic expressions. Also, although the proposed model can model and generate a wide range of realistic expressions, strict definition of the extremeness of each target expression is not possible. This is because the deformation motion labels values range between (0,1), with 0 denoting the neutral expression and 1 representing the apex of each expression. However, candidates in the 4DFAB database express their emotions at different levels of intensity, ranging from moderate to extreme deformations, while in the proposed annotation, apex expressions are always assigned a value of 1. One of the future tasks is to take into consideration the range of each deformation and enhance the proposed model with a user-customizable parameter that could be utilized to control the level of expression. In addition, in order to synthesize realistic facial animation, it is essential to model facial wrinkles along with

shape deformations for each expression. Thus, an attempt will be made to generalize facial expression animation to include both shape and texture by extrapolating the proposed model to texture predictions.

4.6 Conclusion

In this chapter, the first generative model is proposed to synthesize 3D dynamic facial expressions from a still neutral 3D mesh. The model captures both local and global expression deformations using graph-based upsampling and convolution operators. Given a neutral expression mesh of a subject and a time signal that conditions the expected expression motion, the model generates dynamic facial expressions for the same subject that respect the time conditions and the anticipated expression. This is achieved by compressing the expression timeline information to a latent space using RNNs and decoding the respective latent representations with spiral mesh convolutions. The proposed method models the animation of each expression and deforms the neutral face of each subject according to a desired motion. Both expression and motion can be fully defined by the user. Results demonstrate that the proposed method outperforms expression blendshapes and creates motion-consistent deformations, validated both qualitatively and quantitatively. Additionally, an assessment was conducted to determine whether the generated expressions can be correctly classified and identified by a classifier trained on the same dataset. Classification results support the qualitative findings, showing that the generated data can be similarly classified compared to the ones created by the blendshapes model. In summary, the proposed model is the first attempt to synthesize realistic and high-quality facial expressions from a single neutral face input.

CHAPTER 5

3D POINT CLOUD SIMPLIFICATION

Contents

The recent advances in 3D sensing technology have made possible the capture of point clouds in significantly high resolution. However, increased detail usually comes at the expense of high storage, as well as computational costs in terms of processing and visualization operations. Mesh and Point Cloud simplification methods aim to reduce the complexity of 3D models while retaining visual quality and relevant salient features. Traditional simplification techniques usually rely on solving a time-consuming optimization problem, hence they are impractical for large-scale datasets. In an attempt to alleviate the computational burden, a fast point cloud simplification method is proposed, which involves learning to sample salient points. The proposed method relies on a graph neural network architecture trained to select an arbitrary, user-defined, number of points according to their latent encodings and re-arrange their positions so as to minimize the visual perception error. The approach is extensively evaluated on various datasets using several perceptual metrics. Importantly, the proposed method is able to generalize to out-of-distribution shapes, hence demonstrating zero-shot capabilities.

5.1 Introduction

The progress in sensing technologies has significantly expedited the 3D data acquisition pipelines which in turn has increased the availability of large and diverse 3D datasets. With a single 3D sensing device [168], one can capture a target surface and represent it as a 3D object, with point clouds and meshes being the most popular representations. Several applications, ranging from virtual reality and 3D avatar generation [169, 23] to 3D printing and digitization of cultural heritage [170], depend from such representations. However, a 3D capturing device generates thousands of points per second, making processing, visualization and storage of captured 3D objects a computationally daunting task. Often, raw point sets contain an enormous amount of redundant and putatively noisy points with low visual perceptual importance, which results into an unnecessary increase in the storage costs. Thus real-time processing, rendering and editing applications require the development of efficient simplification methods that discard excessive details and reduce the size of the object, while preserving their significant visual characteristics. In contrast to sampling methods that aim to preserve the overall point cloud structure, simplification methods attempt to solve the non-trival task of preserving the semantics of the input objects [171]. Visual semantics refer to the salient features of the object that mostly correlate with human perception and determine its visual appearance in terms of curvature and roughness characteristics of the shape [172, 173]. As can be easily observed in an indicative case shown in Figure 5.1, effortless sampling techniques, such as Farthest Point Sampling (FPS) or uniform sampling, can easily preserve the structure of a point cloud. However, the preservation of perceptually visual characteristics, especially when combined with structural preservation, remains a challenging task. Traditional simplification methods manage to retain the structural and the salient characteristics of large point clouds [58, 174, 175] by constructing a point importance queue that sorts points according to their scores at every iteration. However, apart from being very time demanding, such optimizations are non-convex with increased computational requirements and can not be generalized to different topologies. On the contrary, an end-to-end differentiable neural-based simplification method could leverage the parallel processing of neural networks and simplify batches of point clouds in one pass [24].

In this Chapter, the limitations of the literature on the task of point cloud simplification are tackled, and the first learnable point cloud simplification method is proposed. Motivated by the methods that highlight and demonstrate the importance of perceptual saliency, a method is proposed that preserves both the salient features as well as the overall structure of the input. The method can be used for real-time point cloud simplification without any prior surface reconstruction. The proposed method is fully differentiable,

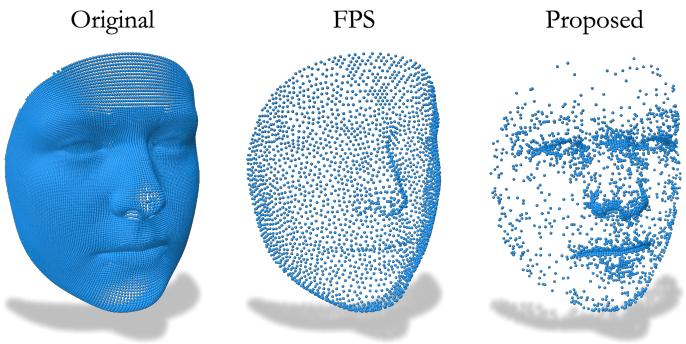


Figure 5.1: Point cloud simplified using FPS (left) smooths out facial characteristics of the input whereas the proposed method preserves salient features of the input (right).

allowing for direct integration into any learnable pipeline without modification. Additionally, a fast latent space clustering using FPS is introduced, which can benefit various fields such as graph partitioning, generative models, and shape interpolation. The clustering method serves as a fast and non-iterative alternative to differentiable clustering, guided by the loss function for the selection of cluster centers. The limitations of popular distance metrics, like the Chamfer distance, to capture salient details of the simplified models are highlighted. Several evaluation criteria well-suited for simplification tasks are proposed. The proposed method is extensively evaluated in a series of wide-ranging experiments.

5.2 Related Work

Point Cloud Simplification and Sampling

Similar to mesh simplification, iterative point selection and clustering techniques have also been proposed for point clouds [174, 176, 177, 178, 175]. In particular, *point cloud simplification* can be addressed either via mesh simplification where the points are fitted to a surface and then simplified using traditional mesh simplification objectives [179, 180], or via direct optimization on the point cloud where the points are selected and decimated according to their estimated local properties [174, 176, 181, 178, 20]. However, similar to mesh simplification, computationally expensive iterative optimization is needed, making them inefficient for large scale point clouds. The point cloud simplification methodology presented in this Chapter attempts to address and overcome the inefficiencies of the afore-

mentioned approaches using a learnable alternative that works with arbitrary, user-defined decimation factors.

In a different line of research, *sampling methods* rely on a point selection scheme that focuses on retaining either the overall structure of the object or specific components of the input. A huge difference between simplification and sampling is founded upon their point selection perspective. Sampling methods are usually utilized for hierarchical learning [22] in contrast to simplification methods that attempt to preserve as much of the visual appearance of the input even at very low resolutions. Farthest Point Sampling (FPS) [182], along with several modification of it, remains the most popular sampling choice and has been widely used as a building block in deep learning pipelines [20, 22, 183]. Nevertheless, as experimentally shown, FPS directly from the input xyz-space can not preserve sharp details of the input and thus is not suitable for simplification tasks. Recently, several methods [184, 185] have been proposed as a learnable alternative for task-driven sampling, optimized for downstream tasks. However, they require the input point clouds to have the same size which limits their usage to datasets with arbitrary topologies. In addition, they explicitly generate the sampled output using linear layers which is not scalable to large point clouds. Although the learnable sampling methods are closely related to the proposed method, they only sample point clouds in a task-driven manner and as a result, the preservation of the high frequency details of the point cloud is not ensured.

Assessment of Perceptual Visual Quality

Processes such as simplification, lossy compression and watermarking inevitably introduce distortion to the 3D objects. Measuring the visual cost in rendered data is a long studied problem [186, 187]. Inspired by Image Quality Assessment measures, the objective of Perceptual Visual Quality (PVQ) assessment is to measure the distortion of an object in terms directly correlated with the Human Perceptual System (HPS). Several methods have been proposed, acting directly on 3D positions, to measure the PVQ using Laplacian distances [188], curvature statistics [189, 190, 191], dihedral angles [192] or per vertex roughness [187]. Several studies [193, 194, 195, 196] utilized crowdsourcing platforms and user subjective assessments to identify the most relevant geometric attributes that mostly correlate with human perception. The findings demonstrated that curvature related features along with dihedral angles and roughness indicate strong similarity with the HPS. In this Chapter, curvature-related losses and quality measures were utilized to train and assess the performance of the proposed model. These measures are referred to as perceptual measures in this context.

5.3 Method

5.3.1 Preliminaries: Point Curvature Estimation

Calculating the local surface properties of an unstructured point cloud is a non-trivial problem. As demonstrated in [55, 174], covariance analysis can be an intuitive estimator of the surface normals and curvature. In particular, considering a neighborhood $\mathcal{N}_i$ around the point $\mathbf{p}_i \in \mathbb{R}^3$ the covariance matrix can be defined as:

$$C = \begin{bmatrix} \mathbf{p}_{i_1} - \mathbf{p}_i \\ \mathbf{p}_{i_2} - \mathbf{p}_i \\ \vdots \\ \mathbf{p}_{i_k} - \mathbf{p}_i \end{bmatrix}^T \cdot \begin{bmatrix} \mathbf{p}_{i_1} - \mathbf{p}_i \\ \mathbf{p}_{i_2} - \mathbf{p}_i \\ \vdots \\ \mathbf{p}_{i_k} - \mathbf{p}_i \end{bmatrix} \in \mathbb{R}^{|\mathcal{N}_i| \times |\mathcal{N}_i|} \tag{5.1}$$

By solving the eigendecomposition of the covariance matrix $\mathbf{C}$, the eigenvectors corresponding to the principal eigenvalues define an orthogonal frame at point $\mathbf{p}_i$. The eigenvalues λ_i measure the variation along the axis defined by their corresponding eigenvector. Intuitively, the eigenvectors that correspond to the largest eigenvalues span the tangent plane at point $\mathbf{p}_i$, whereas the eigenvector corresponding to the smallest eigenvalue can be used to approximate the surface normal $\mathbf{n}_i$. Thus, given that the smallest eigenvalue λ_0 measures the deviation of point $\mathbf{p}_i$ from the surface, it can be used as an estimate of point curvature. As shown in [174], the point curvature can be defined as:

$$\kappa(\mathbf{p}_i) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}, \quad \lambda_0 < \lambda_1 < \lambda_2 \tag{5.2}$$

as the local curvature estimate at point $\mathbf{p}_i$ which is ideal for tasks such as point simplification. Using the previously estimated curvature at point $\mathbf{p}_i$ the mean curvature can be estimated as the Gaussian weighted average of the curvatures around the neighborhood $\mathcal{N}_i$:

$$\bar{\mathcal{K}}(\mathbf{p}_i) = \frac{\sum\limits_{j \in \mathcal{N}_i} \kappa(\mathbf{p}_j) \exp\left(-\|\mathbf{p}_j - \mathbf{p}_i\|^2 / h\right)}{\sum\limits_{j \in \mathcal{N}_i} \exp\left(-\|\mathbf{p}_j - \mathbf{p}_i\|^2 / h\right)} \tag{5.3}$$

where h is a constant defining the neighborhood radius. Finally, an estimation of the roughness can be defined as the difference between curvature and the mean curvature at point $\mathbf{p}_i$ as: $\mathbb{R}(\mathbf{p}_i) = |\kappa(\mathbf{p}_i) - \bar{\mathcal{K}}(\mathbf{p}_i)|$

5.3.2 Model

The main building block of the proposed architecture is a graph neural network that receives at its input a point cloud (or a mesh) $\mathcal{P}_1$ with N points $\mathbf{p}_i$ and outputs a simplified version $\mathcal{P}_2$ with M points, $M << N$. It is important to note that the simplified point cloud $\mathcal{P}_2$ does not need to be a subset of the original point set $\mathcal{P}_1$. The proposed model is composed by three modules: the *Projection Network*, the *Point Selector* and the *Refinement Network*. Figure 6.2 illustrates the architecture of the proposed method.

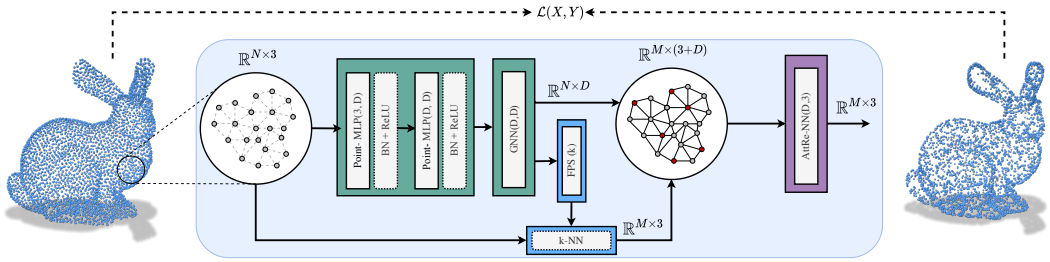


Figure 5.2: Overview of the proposed method. Initially a point cloud (or a mesh) is passed through a projection network (green) and embedded to a higher dimensional latent space. FPS is used to select points from the set of latent representations (blue) that can be conceived as cluster centers of the input. Finally, a k-NN graph is constructed between the cluster centers and the input points that is used to modify their positions using the refinement layer (purple).

Projection Network and Point Selector: In this chapter, sampling is formulated as a clustering problem. The goal is to cluster points that share similar perceptual and structural features and represent the simplified point cloud using the cluster centers. To achieve this, a *Projector Network* is designed to map (x, y, z) coordinates to a high-dimensional space, where points with similar features are close in the latent space. Instead of sampling directly from the Euclidean input space, the objective is to sample points that are embedded in a high-dimensional latent space, capturing the perceptual characteristics of the input. Clustering the latent space results in clusters with latent vectors of points that share similar perceptual characteristics.

Based on the observations that Farthest Point Sampling (FPS) provides a simple and intuitive technique to select points covering the point cloud structure [22], a sampling module is built on top of this sampling strategy, where points are sampled from a high-dimensional space instead of the input *xyz*-space. Although any clustering algorithm could be adequate, the FPS module is utilized as it sufficiently covers the input space without solving any optimization problem. Using this formulation, it is possible to interfere with

the selection process and transform it into a learnable module that is trained to select point embeddings that cover the perceptual latent space, allowing the preservation of both structural and perceptual salient features of the input.

Projector Network comprises of a multi-layer perceptron (MLP) applied to each point independently, followed by a Graph Neural Network (GNN) that captures the local geometric properties around each point. The update rule of the GNN layer is the following:

$$\mathbf{f}_i' = \mathbf{W}_c \mathbf{f}_i + \frac{1}{\mathcal{N}_i} \sum_{j \in \mathcal{N}_i} \mathbf{W}_n \mathbf{f}_j \tag{5.4}$$

where $\mathbf{f}_i$ denotes the output of the shared point-wise MLP for point $\mathbf{p}_i$ and $\mathbf{W}_c, \mathbf{W}_n$ represent learnable projection matrices. The connectivity between points can be given either by the mesh triangulation or by a k-nn query in the input space (a small neighborhood of k =7 was used as in [197]). Following the Projector Network, ***Point Selector*** utilizes FPS to select points, i.e. cluster centers, based on their latent representations, in order to cover the latent space. Given the cluster centers selected by FPS, a k-nn graph is constructed that connects the center points with their k-nearest neighbours, based on their 3D positions.

Attention-based Refinement Layer: Cluster centers, their neighboring point positions along with their respective embeddings from the projection networks are passed to the attention-based refinement layer (AttRef) that modifies the positions of the cluster centers. This layer can be considered as a rectification step that given a neighborhood and its corresponding latent features, displaces the cluster center points in order to minimize the visual perceptual error. Given that the latent embeddings of each point can be considered as its local descriptor, the refinement layer generates the new positions based on the vertex displacements along with the neighborhood local descriptors. The final positions of the points as predicted by *AttRef* are defined as follows:

$$\mathbf{p}_{c_i}' = \mathbf{p}_{c_i} + \gamma \left(\frac{1}{\mathcal{N}_{c_i}} \sum_{j \in \mathcal{N}_{c_i}} \alpha_{ij} \phi([\mathbf{f}_j \| \mathbf{p}_j - \mathbf{p}_{c_i}]) \right) \tag{5.5}$$

where γ and ϕ are MLPs, $\mathcal{N}_{c_i}$ the k-nearest neighbors of point $\mathbf{p}_{c_i}$ (k = 15 was used), $\mathbf{f}_j$ the latent features of point $\mathbf{p}_j$ and α_{ij} the attention coefficients between center $\mathbf{p}_{c_i}$ and point $\mathbf{p}_j$. The attention coefficients α_{ij} are computed using scaled dot-product [198], i.e. $\alpha_{ij} = \text{softmax}\left(\frac{\theta_q(\mathbf{f}_j)^T \theta_k(\mathbf{f}_i)}{\sqrt{d}}\right)$, where θ_q, θ_k are linear transformations mapping features $\mathbf{f}$ to a d-dimensional space.

5.3.3 Loss Function

The selection of the loss function to be optimized is crucial for the task of simplification since a balance is required between the preservation of the object's structure and its underlying salient features. A major barrier of most common distance metrics is the uniform weighting of points that can not reflect the perceptual differences between objects. As shown in many studies [199, 200, 201] the commonly used Chamfer distance (CD) between two point sets $\mathcal{P}_1, \mathcal{P}_2$ defined as:

$$d_{\mathcal{P}_1,\mathcal{P}_2} = \sum_{\mathbf{x}\in\mathcal{P}_1} \min_{\mathbf{y}\in\mathcal{P}_2} \|\mathbf{x}-\mathbf{y}\|^2 + \sum_{y\in\mathcal{P}_2} \min_{\mathbf{x}\in\mathcal{P}_1} \|\mathbf{x}-\mathbf{y}\|^2 \tag{5.6}$$

can only describe the overall surface structure similarity between the two sets without taking into account the high frequency details of each point cloud. Figure 5.1 illustrates an example of such case. Similarly, the point to surface distance between points of a set $\mathcal{P}$ and a surface $\mathcal{M}$ as well as the Hausdorff distance can not preserve salient points of the object rather than its global appearance. To train the proposed model it is essential to devise a loss function that preserves both the salient features along with the structure of the point cloud.

Adaptive Chamfer Distance: As can be easily observed, the first term of Eq. (5.6) measures the preservation of the overall structure of $\mathcal{P}_1$ by $\mathcal{P}_2$, in a uniform way. To break the uniformity of the first term of CD, a weighting factor w_x is introduced in equation 5.7 that penalizes the distances between the two sets at the points with high salient features, ensuring that they will be preserved in the simplified point cloud. The modified adaptive Chamfer distance is defined as follows:

$$d^{Adapt}_{\mathcal{P}_1,\mathcal{P}_2} = \sum_{\mathbf{x}\in\mathcal{P}_1} w_{\bar{\mathcal{K}}(\mathbf{x})} \min_{\mathbf{y}\in\mathcal{P}_2} \|\mathbf{x}-\mathbf{y}\|^2 + \sum_{\mathbf{y}\in\mathcal{P}_2} \min_{x\in\mathcal{P}_1} \|\mathbf{x}-\mathbf{y}\|^2 \tag{5.7}$$

where $\mathcal{P}_1$ denotes the initial point cloud, $\mathcal{P}_2$ the simplified one, and $w_{\bar{\mathcal{K}}(\mathbf{x})}$ a weighting factor proportional to the mean curvature $\bar{\mathcal{K}}$ at point $\mathbf{x}$[1] This weighting factor is only applied to the first term of equation (5.6) to retain salient points of $\mathcal{P}_1$. The second term of the equation is not weighted to avoid the optimization process from getting trapped at local minima.

Curvature Preservation: In addition to the adaptive Chamfer distance, the use of a loss term reinforces the selection of high curvature points from the input. To quantify the

[1]The weights w_x are defined using the sigmoid function applied to the normalized curvatures, divided by a temperature scalar $\tau = 10$, in order to amplify high curvature values.

preservation of salient features of the input, an error is introduced to measure the average point-wise curvature distance between the two point clouds:

$$\mathcal{E}_c = \left(\frac{1}{|\mathcal{P}_1|} \sum_{x \in \mathcal{P}_1} \|\bar{\mathcal{K}}_1(\mathbf{x}) - \bar{\mathcal{K}}_2(\mathrm{NN}(\mathbf{x}, \mathcal{P}_2))\|^2 \right)^{1/2} \tag{5.8}$$

where $\mathrm{NN}(\mathbf{x}, \mathcal{P}_2)$ denotes the nearest neighbour of $\mathbf{x}$ in set $\mathcal{P}_2$, and $\bar{\mathcal{K}}(\cdot)$ the mean curvature. This error is referred as Curvature Error (CE).

Overall Objective: A combination of the two aforementioned losses was used as the total objective to be minimized:

$$\mathcal{L}(\mathcal{P}_1, \mathcal{P}_2) = d^{Adapt}_{\mathcal{P}_1, \mathcal{P}_2} + \lambda \mathcal{E}_c \tag{5.9}$$

where λ is used as a scaling factor set to 0.1. The first term ensures that the selected points cover the surface of the input, while the latter enforces the selection of high curvature points.

5.4 Evaluation Criteria

To assess the performance of the simplified models generated by the proposed method in terms of visual perception, several metrics were defined that measure the similarity between the two point cloud models.

Roughness Preservation: Roughness describes the deviation of a point from the surface defined by its neighbours and has been identified as a salient feature in many visual perception studies [172, 202]. Similar to the curvature preservation loss, the roughness preservation error was calculated by substituting the curvature values with roughness in eq (5.8). This error is referred as RE .

Point Cloud Structural Distortion Measure: Additionally to curvature and roughness preservation metrics, the structural similarity score between the two point clouds is also calculated, which has been shown to highly correlate with human perception [203]. In particular, the point cloud Structural Distortion Measure (SDM) can be defined as:

$$D(\mathcal{P}_1, \mathcal{P}_2) = \frac{\alpha \mathcal{L}(\mathbf{p}_i, \hat{\mathbf{p}}_i) + \beta \mathcal{C}(\mathbf{p}_i, \hat{\mathbf{p}}_i) + \gamma \mathcal{S}(\mathbf{p}_i, \hat{\mathbf{p}}_i)}{\alpha + \beta + \gamma} \tag{5.10}$$

$$\mathcal{L}(\mathbf{p}_i, \hat{\mathbf{p}}_i) = \frac{||\bar{\mathcal{K}_1}(\mathbf{p}_i) - \bar{\mathcal{K}_2}(\hat{\mathbf{p}}_i)||}{\max(\bar{\mathcal{K}_1}(\mathbf{p}_i), \bar{\mathcal{K}_2}(\hat{\mathbf{p}}_i))} \tag{5.11}$$

$$C(\mathbf{p}_i, \hat{\mathbf{p}}_i) = \frac{||\sigma_{\bar{\mathcal{K}_1}}(\mathbf{p}_i) - \sigma_{\bar{\mathcal{K}_2}}(\hat{\mathbf{p}}_i)||}{\max(\bar{\mathcal{K}_1}(\mathbf{p}_i), \bar{\mathcal{K}_2}(\hat{\mathbf{p}}_i))} \tag{5.12}$$

$$\mathcal{S}(\mathbf{p}_i, \hat{\mathbf{p}}_i) = \frac{||\sigma_{\bar{\mathcal{K}_1}}(\mathbf{p}_i)\sigma_{\bar{\mathcal{K}_2}}(\hat{\mathbf{p}}_i) - \sigma_{\bar{\mathcal{K}}_{12}}(\mathbf{p}_i, \hat{\mathbf{p}}_i)^2||}{\sigma_{\bar{\mathcal{K}_1}}(\mathbf{p}_i)\sigma_{\bar{\mathcal{K}_2}}(\hat{\mathbf{p}}_i)} \tag{5.13}$$

where $\mathcal{K}_1$, $\mathcal{K}_2, \sigma_{\bar{\mathcal{K}_1}}, \sigma_{\bar{\mathcal{K}_2}}, \sigma_{\bar{\mathcal{K}}_{12}}(\mathbf{p}_i, \hat{\mathbf{p}}_i)$ are the mean, the gaussian-weighted standard deviation and the covariance of the curvatures for point p_i in $\mathcal{P}_1$ and its corresponding point $\hat{\mathbf{p}}_i$ in $\mathcal{P}_2$, respectively. The correspondence between the two point clouds is established by using the 1-nearest neighbor for each point. The global similarity score is obtained using *Minkowski pooling* as suggested in [190].

Normals Consistency: Point normals are highly related to visual appearance as indicators of sharp and smooth areas. The consistency of normals' orientations between the two models was measured using the cosine similarity loss:

$$\mathcal{E}_n = \frac{1}{|\mathcal{P}_1|} \sum_{\substack{\mathbf{x}\in\mathcal{P}_1 \\ \mathbf{y}\in NN(x,\mathcal{P}_2)}} 1 - \frac{\mathbf{n_x}\cdot\mathbf{n_y}}{\|\mathbf{n_x}\|\|\mathbf{n_y}\|} + \frac{1}{|\mathcal{P}_2|} \sum_{\substack{\mathbf{y}\in\mathcal{P}_2 \\ \mathbf{x}\in NN(y,\mathcal{P}_1)}} 1 - \frac{\mathbf{n_x}\cdot\mathbf{n_y}}{\|\mathbf{n_x}\|\|\mathbf{n_y}\|} \tag{5.14}$$

where $\mathbf{n_x}$ denotes the normal at point x and $NN(\mathbf{x}, \mathcal{P}_2)$ the nearest neighbour of $\mathbf{x}$ in set $\mathcal{P}_2$, calculated as described in Section 5.3.1.

5.5 Experiments

In this section extensive evaluation of the proposed method with both quantitative and qualitative experiments is reported.

Baselines

The proposed approach was compared against several sampling and simplification methods including: uniform sampling (random), FPS, PointASNL adaptive sampling method [183], quadric error metric (QEM) simplification [58], spectral mesh simplification [68], feature

preserving point cloud simplification [175] along with a top curvature points sampling (TCP) where the top-k curvature points are selected from the input point cloud. Comparison with recent simplification methods such as [204, 69], which rely on the eigendecomposition of the Laplacian matrix, was not possible due to their overwhelmingly large processing run-time and memory consumption (approximately 15 minutes for a mesh with approximately 15K points).

Datasets

The proposed method was evaluated using several publicly available 3D datasets, with different characteristics. The simplification benchmark TOSCA [205] dataset comprises 80 synthetic high-resolution meshes with 9 different deformable objects. It is an excellent candidate to assess feature-preserving simplification, since most of its meshes are non-smooth consisting of high curvature regions. Additionally, the popular ModelNet10 dataset [18] was used along with the fixed topology high-resolution MeIn3D face dataset [206]. All datasets used were randomly split in 80%-20% train-test sets, taking care that none of the identities/shapes used for training are present in the respective test set.

Evaluation Criteria

The quality of the simplified point clouds is quantitatively evaluated in three ways. Firstly, low-level structural and perceptual measures are measured, as described in Section 5.4. Additionally, a pre-trained objective classifier is used to measure the preservation of high-level semantics, and a user-study is conducted to assess the perceptual similarity between the input and the simplified models. Furthermore, ablation studies are performed to evaluate the importance of the essential components of the proposed method. Finally, the performance of the method was also assessed under noisy conditions and on real-world scans.

5.5.1 Point Cloud Simplification

In this section, the simplification performance of the proposed method is presented. For each dataset, both structural (i.e. CD, NC) as well as perceptual metrics (i.e. CE, RE, SDM) were reported for the proposed and the baseline methods. The figures reported in Table 5.1 indicate the superiority of the proposed method to maintain the perceptual features of the input (i.e. low SDM) without sacrificing the overall structure (i.e. low CD)

Table 5.1: Simplification performance tested on TOSCA (top), ModelNet10 (middle) and MeIn3d (bottom) datasets for *low simplification ratios* ($N_s/N_{org} < 0.2$) . Best approaches are highlighted in **bold** and second best in **red**. For generalization comparison we trained the three models "Proposed-*MeIn3D*", "Proposed-*ModelNet*", "Proposed-*TOSCA*" on MeIn3D, ModelNet and TOSCA datasets respectively.

	TOSCA											
	N_s/N_{org} = 0.2				N_s/N_{org} =0.1				N_s/N_{org} =0.05			
Method	CD	NC	RE($\times10^{-4}$)	SDM($\times10^{-4}$)	CD	NC	RE($\times10^{-4}$)	SDM($\times10^{-4}$)	CD	NC	RE($\times10^{-4}$)	SDM($\times10^{-4}$)
Random	1.63	0.312	4.45	6.07	3.35	0.342	4.91	10.7	6.68	0.369	5.71	19.2
TCP	51.3	0.625	4.99	9.52	129.4	0.732	6.42	17.8	172.5	0.793	6.20	32.4
FPS	**0.81**	0.307	4.71	**5.13**	**1.93**	0.341	4.82	9.64	**3.94**	**0.321**	5.56	18.3
QEM	1.35	**0.291**	**4.01**	5.36	2.64	**0.310**	4.79	10.4	**4.77**	0.338	5.53	18.4
Liu *et al.* [68]	2.17	0.358	4.39	5.39	3.12	0.331	4.96	10.4	5.62	0.441	5.96	18.5
Qi *et al.* [175]	2.49	0.303	4.45	7.37	3.46	0.353	4.51	13.18	6.15	0.372	5.34	23.18
Yan *et al.* [183]	1.17	0.301	4.27	5.41	2.54	0.321	4.48	9.51	5.14	0.357	5.27	18.1
Proposed MeIn3D	1.14	0.293	4.15	5.64	2.53	0.313	**4.47**	**8.15**	5.36	0.364	5.01	17.7
Proposed ModelNet	1.15	0.310	**4.01**	5.53	2.51	0.312	4.81	9.72	5.19	0.341	**4.99**	**17.4**
Proposed TOSCA	**1.12**	**0.290**	**3.91**	**5.01**	**2.45**	**0.307**	**4.41**	**7.84**	4.93	**0.333**	**4.93**	**16.5**

	ModelNet											
	N_s/N_{org} = 0.2				N_s/N_{org} =0.1				N_s/N_{org} =0.05			
Method	CD ($\times10^{-4}$)	NC	RE($\times10^{-5}$)	SDM($\times10^{-3}$)	CD($\times10^{-5}$)	NC	RE($\times10^{-5}$)	SDM($\times10^{-3}$)	CD($\times10^{-4}$)	NC	RE($\times10^{-5}$)	SDM($\times10^{-3}$)
Random	8.01	0.568	5.91	2.83	20.4	0.655	6.19	4.92	41.02	0.793	6.57	8.19
TCP	197.3	0.898	7.25	3.87	403.1	0.937	7.84	7.11	611.6	0.952	7.01	12.81
FPS	**3.12**	**0.505**	6.05	2.74	**7.56**	0.641	6.39	4.81	**16.01**	0.744	6.48	8.38
QEM	3.45	**0.513**	5.94	3.01	9.45	0.625	6.13	5.19	21.43	0.724	6.25	9.12
Liu *et al.* [68]	4.21	0.537	5.99	3.05	10.32	0.632	6.75	5.32	21.54	0.792	6.52	9.44
Qi *et al.* [175]	5.64	0.515	6.03	3.47	10.97	0.654	6.54	5.71	26.37	0.745	6.39	9.17
Yan *et al.* [183]	6.28	0.514	5.87	2.89	11.08	0.643	6.29	5.04	20.69	0.428	6.37	8.61
Proposed-MeIn3D	4.02	0.531	5.93	2.86	29.31	0.610	6.08	4.76	45.12	0.701	6.33	8.02
Proposed-ModelNet	**3.32**	0.515	**5.79**	**2.68**	**8.24**	**0.606**	**6.06**	**4.61**	**17.24**	**0.696**	**6.25**	**7.92**
Proposed-TOSCA	4.35	0.523	**5.77**	**2.72**	9.42	**0.603**	**5.91**	**4.64**	22.18	**0.688**	**6.04**	**7.96**

	MeIn3D											
	N_s/N_{org} = 0.2				N_s/N_{org} =0.1				N_s/N_{org} =0.05			
Method	CD ($\times10^{-4}$)	NC	RE($\times10^{-5}$)	SDM($\times10^{-3}$)	CD($\times10^{-5}$)	NC	RE($\times10^{-5}$)	SDM($\times10^{-3}$)	CD($\times10^{-4}$)	NC	RE($\times10^{-5}$)	SDM($\times10^{-3}$)
Random	1.42	0.198	4.15	2.81	3.46	0.313	6.73	5.92	5.52	0.481	7.05	12.4
TCP	158.3	0.801	3.46	3.73	421.1	0.910	6.02	7.38	556.0	0.934	11.87	14.2
FPS	**1.12**	**0.121**	3.64	2.96	**1.93**	0.195	6.29	5.98	**3.45**	0.484	7.43	11.8
QEM	2.01	0.185	4.53	3.01	2.52	0.198	6.31	5.71	3.65	**0.331**	8.13	11.3
Liu *et al.* [68]	2.92	0.215	4.92	3.12	3.14	0.199	6.67	5.92	3.88	0.392	8.46	11.9
Qi *et al.* [175]	3.12	0.193	4.74	3.45	3.55	0.211	6.51	5.95	4.07	0.405	8.22	12.6
Yan *et al.* [183]	2.75	0.193	4.70	2.95	3.05	0.205	6.34	5.41	3.76	0.374	9.17	11.4
Proposed-MeIn3D	**1.24**	**0.128**	**3.15**	**2.30**	**2.01**	**0.192**	**5.69**	**4.91**	**3.25**	**0.305**	**6.47**	**10.6**
Proposed-ModelNet	1.75	0.189	3.65	2.45	3.23	0.196	**5.73**	5.10	4.02	0.369	7.02	10.9
Proposed-TOSCA	1.54	0.168	**3.29**	**2.41**	2.32	**0.194**	5.98	**5.06**	3.82	0.342	**6.49**	**10.8**

of the shape at three indicative simplification ratios. In contrast to TPC method where the selection of high curvature points leads to an increased Chamfer distance, the proposed method achieves a fair balance between structure (CD and NC) and saliency (SDM and RE). Additionally, the proposed method exhibits lower perceptual error (SDM) compared to FPS and PointASNL [183] methods that sample points directly from the xyz-space. This may be also observed in Figure 5.3 where sampling from the perceptual latent space manages to effectively induce the preservation of the input point cloud details.

A significant result, as it can be observed from Table 5.1 and Figure 6.5, is that the proposed method outperforms recent methods [68, 175, 183] under all metrics, and QEM in terms of perceptual error. Figure 5.5, demonstrates the superiority of the proposed method to remarkably retain the salient points of the input when 1% of the input is retained, in contrast to QEM that performs poorly at coarse areas of high curvature. The selection of salient points of the input is demonstrated in Figure 5.3, where the proposed method favours point selection around the chair's arm, the face's eyes and nose in contrast to points at smooth areas, such as the forehead. Intuitively, smooth areas require only

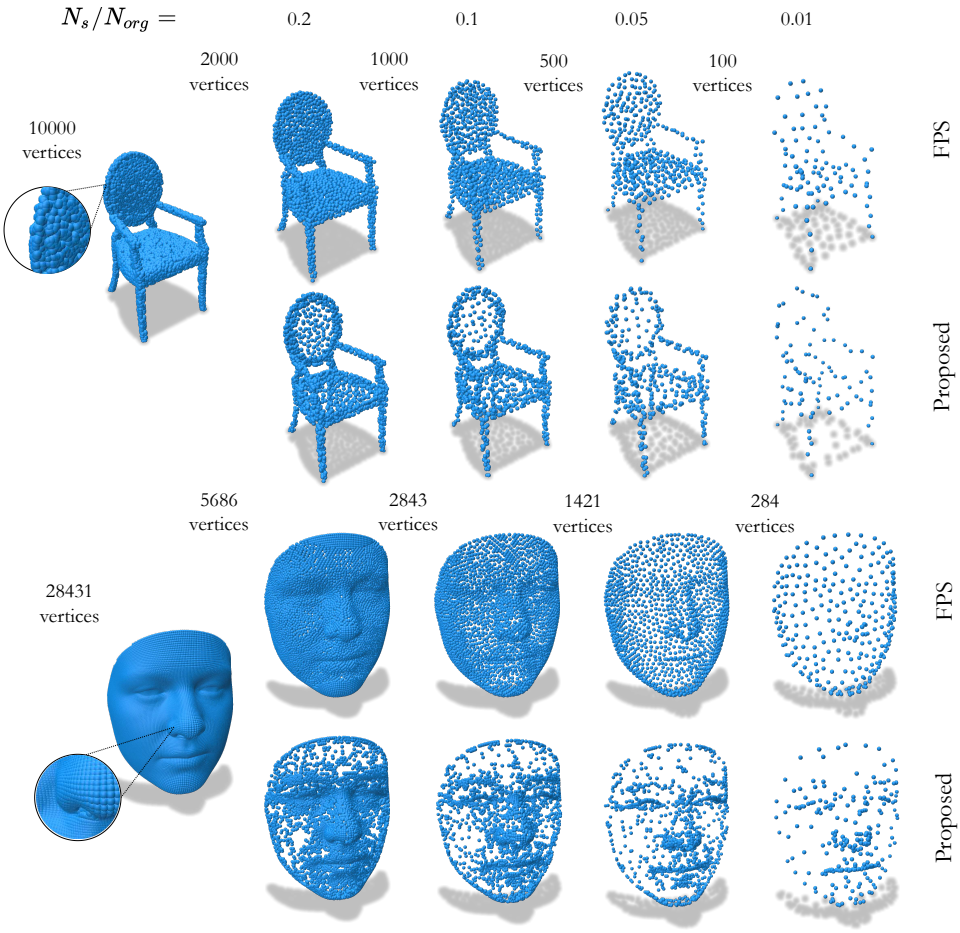


Figure 5.3: Qualitative comparison between FPS (top row) and the proposed (bottom row) methods, at different simplification ratios. Differences between the two methods can be found at coarse and smooth areas, where the proposed model favours the preservation of high-frequency details of the input point cloud.

a few points to describe their associated planes compared to coarse areas that demand many points in order to preserve their curvature.

Experiments were conducted in a cross-dataset generalization scenario, where different datasets were used for training and testing the model. The results of these experiments are shown in the bottom rows of Table 5.1. As it may be observed, it is of interest and of high importance that the proposed model can generalize well to out-of-distribution shapes and topologies indicating that it can be applied directly to any point cloud without fine-tuning, especially when trained with TOSCA or ModelNet dataset. This might be due to

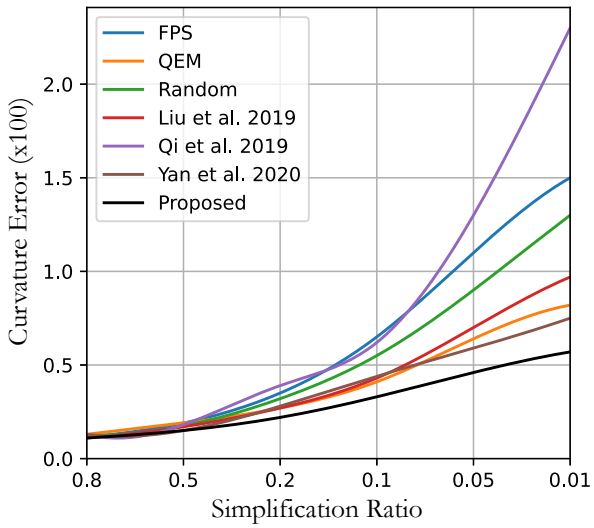


Figure 5.4: Curvature preservation error comparison for the TOSCA dataset at different simplification ratios. Curvature error scales linearly for the proposed method.

the diversity of shapes and topologies as well as the presence of many rough regions at the training sets of these datasets that enforce the model to favour salient features.

Although lower simplification ratios such as the ones presented in 5.1 are more challenging, experiments with high simplification ratios were also performed. Table 5.2 includes the simplification performance of the proposed and the baseline methods on TOSCA, ModelNet and MeIn3D datasets for simplification ratios above 0.3. Note that the method of [68] run out of time when attempted to simplify meshes for large simplification ratios (over 0.4). Importantly, the proposed method outperforms almost all baselines under the perceptual metrics (NC, RE, SDM) and exhibits comparable CD measures with FPS method.

It is also important to note that the performance of the proposed method degrades linearly as the simplification ratio increases, compared to baseline methods such as TCP and Qi *et al.* [175]. Additionally, the proposed method achieves to be the best or the second best performing method under all measures.

5.5.2 Ablation Studies

Loss function: As mentioned in Section 5.3.3, an important component of the proposed simplification framework is the engineering of the curvature guided loss function. In par-

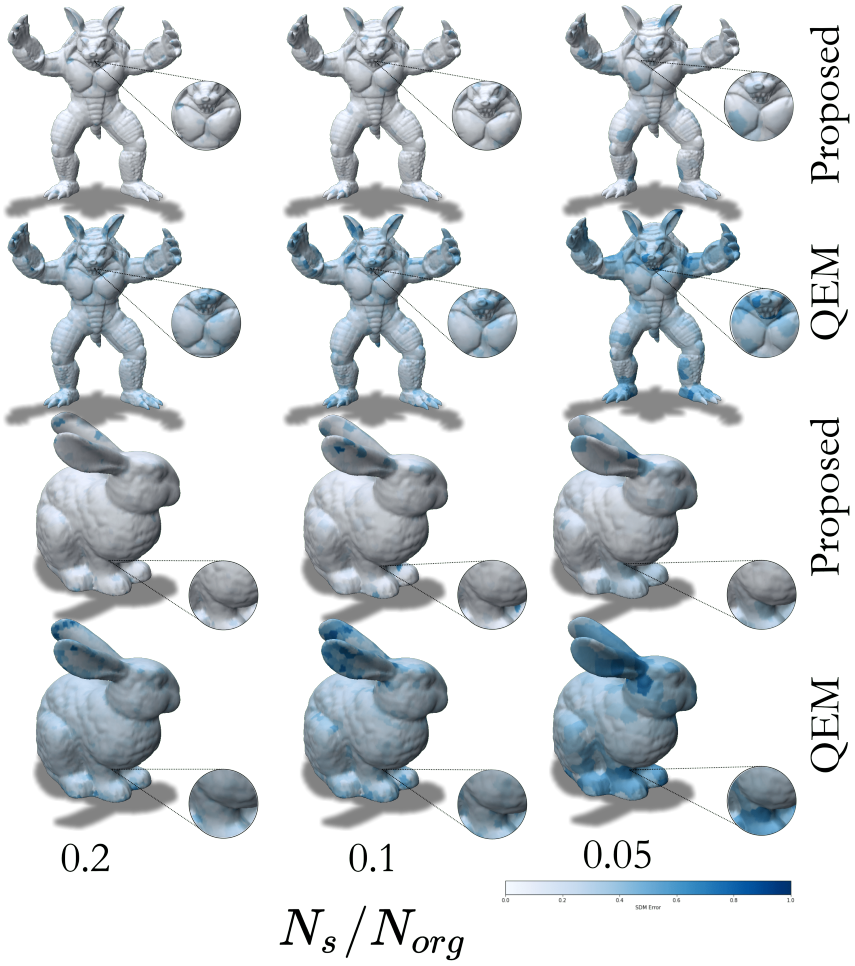


Figure 5.5: Colorcoded curvature error comparison between QEM and the proposed method. Blue color corresponds to larger error.

Table 5.2: Simplification performance tested on TOSCA, ModelNet and MeIn3D datasets for high simplification ratios. Best approaches highlighted are highlighted in **bold** and second best in **red**. The dataset used for training was referred as "Proposed-*Dataset*"

	TOSCA											
	N_s/N_{org} = 0.8				N_s/N_{org} = 0.5				N_s/N_{org} = 0.3			
Method	CD	NC	RE($\times10^{-4}$)	SDM($\times10^{-3}$)	CD	NC	RE($\times10^{-4}$)	SDM($\times10^{-3}$)	CD	NC	RE($\times10^{-4}$)	SDM($\times10^{-3}$)
Random	0.14	**0.093**	2.87	2.43	0.49	**0.106**	3.04	3.03	1.04	**0.225**	3.65	4.33
TCP	1.11	0.147	2.86	2.61	10.0	0.272	3.36	4.05	30.8	0.357	4.19	6.57
FPS	0.09	0.103	**2.85**	2.32	0.29	0.245	**2.97**	2.92	0.67	0.255	**3.52**	4.22
QEM	0.09	0.103	**2.81**	2.33	0.29	0.214	**2.96**	2.91	0.84	**0.248**	3.54	4.27
Liu *et al.* [68]	-	-	-	-	-	-	-	-	1.56	0.384	3.86	4.52
Qi *et al.* [175]	0.10	0.104	2.87	2.47	0.54	0.209	2.98	3.54	1.71	0.253	3.58	5.32
Yan *et al.* [183]	0.28	0.103	2.58	2.51	0.42	0.208	2.98	2.90	0.71	0.250	3.57	4.20
Proposed-MeIn3D	0.05	0.104	2.88	2.30	0.25	0.244	3.06	2.87	0.65	0.255	3.54	**4.07**
Proposed-ModelNet	**0.03**	0.103	2.87	**2.29**	**0.23**	0.211	3.05	**2.83**	**0.64**	0.259	**3.51**	4.08
Proposed-TOSCA	**0.03**	**0.102**	2.86	**2.21**	**0.23**	**0.193**	3.03	**2.79**	**0.63**	0.254	3.55	**4.04**

	ModelNet											
Method	CD($\times10^{-4}$)	NC	RE($\times10^{-4}$)	SDM($\times10^{-3}$)	CD($\times10^{-4}$)	NC	RE($\times10^{-4}$)	SDM($\times10^{-3}$)	CD($\times10^{-4}$)	NC	RE($\times10^{-4}$)	SDM($\times10^{-3}$)
Random	1.74	**0.181**	4.91	1.14	3.13	**0.201**	5.16	1.53	6.01	**0.333**	5.37	1.99
CP	14.01	0.288	5.01	1.12	55.12	0.371	5.98	1.68	117.11	0.527	6.63	2.71
FPS	**0.89**	**0.195**	**4.71**	**1.01**	**1.93**	**0.213**	4.89	**1.35**	**3.02**	**0.352**	5.57	2.08
QEM	1.35	0.211	4.98	1.14	2.84	0.224	5.12	1.48	3.05	0.382	5.57	2.44
Liu *et al.* [68]	-	-	-	-	-	-	-	-				
Qi *et al.* [175]	1.37	0.210	4.97	1.04	3.31	2.31	5.14	1.54	7.04	0.357	5.31	2.17
Yan *et al.* [183]	2.71	0.209	4.31	1.05	3.37	0.235	5.12	1.43	4.64	0.346	5.28	2.11
Proposed-MeIn3D	2.32	0.353	5.12	1.11	2.81	0.365	5.23	1.50	3.72	0.473	5.53	2.15
Proposed-ModelNet	**0.91**	0.207	**4.61**	**0.99**	**1.12**	0.216	**4.72**	**1.28**	**2.74**	0.371	**5.01**	**1.87**
Proposed TOSCA	2.12	0.270	4.82	1.07	2.98	0.283	**4.86**	1.42	4.11	0.401	**5.26**	**2.03**

	MeIn3D											
Method	CD($\times10^{-4}$)	NC	RE($\times10^{-4}$)	SDM($\times10^{-3}$)	CD($\times10^{-4}$)	NC	RE($\times10^{-4}$)	SDM($\times10^{-3}$)	CD($\times10^{-4}$)	NC	RE($\times10^{-4}$)	SDM($\times10^{-3}$)
Random	0.74	0.108	2.46	0.99	1.12	0.120	2.74	1.24	1.26	0.169	3.43	2.31
TCP	12.35	0.211	2.41	0.97	43.06	0.327	**2.52**	1.54	89.24	0.571	**2.89**	2.91
FPS	**0.59**	**0.103**	**2.32**	**0.86**	**0.97**	**0.105**	2.53	**1.15**	**1.05**	**0.108**	3.21	2.28
QEM	0.94	0.112	2.52	1.06	1.36	0.139	2.76	1.44	1.94	0.150	3.53	2.54
Liu *et al.* [68]	-	-	-	-	-	-	-	-	1.99	0.159	3.91	2.83
Qi *et al.* [175]	1.22	0.110	2.77	4.14	1.97	0.131	2.77	1.40	2.15	0.144	3.55	2.44
Yan *et al.* [183]	1.34	0.178	2.45	1.23	1.84	0.129	2.74	1.31	2.07	0.127	3.21	2.19
Proposed MeIn3D	**0.61**	**0.104**	**2.29**	**0.90**	**0.98**	**0.105**	**2.46**	**1.07**	**1.15**	**0.105**	**2.89**	**1.76**
Proposed ModelNet	1.15	0.111	2.41	1.02	1.28	0.123	2.68	1.43	1.59	0.165	2.99	2.09
Proposed TOSCA	1.04	0.106	2.41	1.08	1.21	0.1171	2.63	1.33	1.41	0.149	**2.96**	**1.85**

ticular, Chamfer Distance (CD) assigns an equal importance weight to each point set, neglecting important points of the point cloud. Thus, semantically meaningful points will be assigned with the same penalty as with points at flat smooth areas. In such way, CD will drive the model to generate smooth results that minimize shape reconstruction without taking into account critical identity details of the object. To break this uniformity, the first term of the CD was modified to assign a different weight to each point according to its curvature. In Table 6.3, the performance of the proposed method was reported when trained only with regular CD (Proposed-CD), with adaptive CD (Proposed-ACD), and with both adaptive CD and curvature preservation loss (Proposed-Full). Results reveal that the modified CD exhibits lower perceptual error (CE, RE, SDM) compared to simple CD, while adding a curvature preserving loss (Proposed-Full) further boosts the performance of the model.

Model architecture: Additionally, the importance of the GNN-based point projector and the attention refinement network in the proposed architecture were examined. As shown in Table 6.3, a performance similar to the FPS method is observed when the proposed method is trained without the GNN module (Proposed w/o GNN), with a signi-

ficant increase in perceptual error measures (RE, SDM). This certifies the importance of the GNN-based point projector that, in contrast to linear layers, enables message passing through neighboring points. Finally, an increase in SMD error is also observed to the model trained without the attention refinement module. As expected the use of attention refinement module further improves perceptual preservation since it weights points within the same cluster and moves the cluster centers closer to salient regions to minimize the curvature error.

Table 5.3: Ablation study on loss function and model architecture. Proposed-CD denotes the model trained with CD, Proposed-ACD denotes model trained with adaptive CD and Proposed-Full denotes the model trained with the loss functions introduced in Section 3.3. Proposed-w/o GNN refers to the model trained without the GNN layer in point projector module and Proposed-w/o AttRef refers to the model trained without the attention refinement module.

	N_s/N_{org} = 0.2				N_s/N_{org} =0.1				N_s/N_{org} =0.05			
Method	CD	CE	RE($\times 10^{-4}$)	SDM($\times 10^{-4}$)	CD	CE	RE($\times 10^{-4}$)	SDM($\times 10^{-4}$)	CD	CE	RE($\times 10^{-4}$)	SDM($\times 10^{-4}$)
Proposed-CD	1.12	0.40	3.96	5.52	2.41	0.47	4.43	9.64	4.91	0.58	4.99	18.3
Proposed-ACD	1.15	0.39	4.01	5.51	2.54	0.46	4.42	9.61	4.97	0.56	4.96	17.9
Proposed-w/o GNN	**0.86**	0.32	4.67	5.11	**2.15**	0.35	4.76	9.12	**4.15**	0.36	5.32	18.1
Proposed-w/o AttRef	0.95	0.31	4.21	5.05	2.29	0.31	4.52	8.54	4.51	0.34	5.16	17.3
Proposed-Full	1.12	**0.29**	**3.91**	**5.01**	2.45	**0.30**	**4.41**	**7.84**	4.93	**0.33**	**4.93**	**16.5**

5.5.3 Mesh Simplification

As described in Chapter 2, mesh simplification is a long studied problem that has been tackled only by greedy algorithms. In this section, an alternative method is introduced and proposed to overcome the greedy nature of simplification. This alternative method leverages the simplification technique introduced in Section 5.3. The proposed method, without the need of any particular modification, can be easily extended for the task of mesh simplification when combined with a triangulation algorithm to transform the simplified vertices into a mesh structure. The process unfolds in two steps. Initially, mesh vertices are simplified by treating them as a point cloud but, instead of using a k-nn, the mesh adjacency matrix is utilized in order to determine point connectivity. In a second step, the remaining vertices are re-triangulated using an off-the-self triangulation algorithm such as Ball Pivoting [207]. Other triangulation algorithms, such as Delaunay, alpha shapes or Voronoi diagrams, could be also used but Ball Pivoting algorithm was selected as it produces better results for small point clouds. Figure 5.6 shows visual results for the extension of the proposed simplification method to triangular meshes (for various simplification ratios).

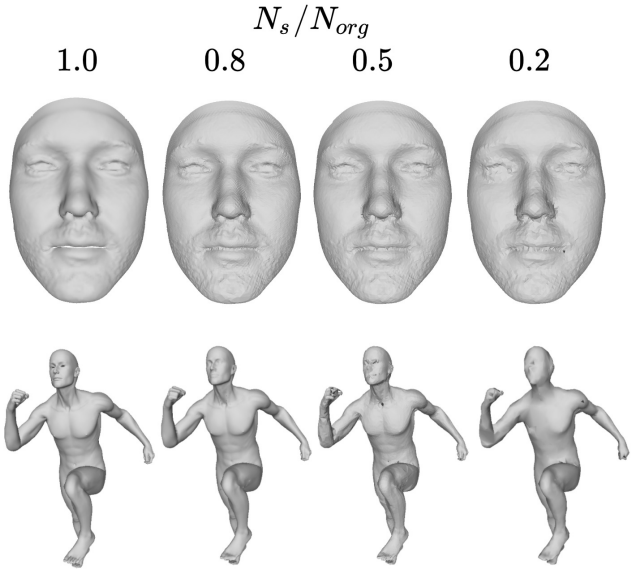


Figure 5.6: Simplified meshes using the proposed method followed by Ball Pivoting Algorithm.

5.5.4 User-Study

To quantify the ability of the proposed method to select points that correlate with human perception a user study was performed, using the paired comparison protocol contrasting the proposed and one baseline method, similar to [140]. In total, 50 participants were specifically asked to evaluate 18 point clouds from different datasets (MeIn3D, ModelNet, TOSCA) and select one of the two simplified point clouds that mostly preserves the perceptual details of the reference one in terms of the overall shape and identity similarity. In average, as shown in Table 5.4, users selected 14 out of the 18 point clouds produced by the proposed method, as the ones preserving most of the visual features.

Table 5.4: User studies results of different methods. Average user preference scores are reported(higher is better, results in %). Best results in bold.

Method	User choice
Proposed vs QEM	**73%**/27%
Proposed vs Liu *et al.* [68]	**78%**/22%
Proposed vs FPS	**71%**/29%
Average	**74%**/26%

5.5.5 Computational Time

Inevitably, in addition to salient point preservation, a proper point cloud simplification method should confront real-time executions. Although time complexity is beyond the scope of this study, the time required for simplifying 80 high-resolution meshes from the TOSCA dataset was assessed . Since FPS, Uniform and TCP baseline methods do not require any significant computations, the proposed method was compared with the popular QEM approach using a highly optimized version from the MeshLab framework [208] and the official implementations of [175] and [68]. It is important to note that the code of the proposed method could be further optimized, using parallel programming. In particular, 80% of the computational time of the proposed method is acquired by k-NN search that could be further optimized whereas FPS takes 17 % and the rest 3% of the runtime is spent on the learnable modules. Figure 5.7 demonstrates that the required mean runtime of the proposed method decreases drastically across all experiments, as the desired simplification increases, requiring just a few seconds to simplify the input to 1% of its original size. In contrast, the methods of [175] and [68] require approximately a minute to simplify a single mesh which makes them impractical.

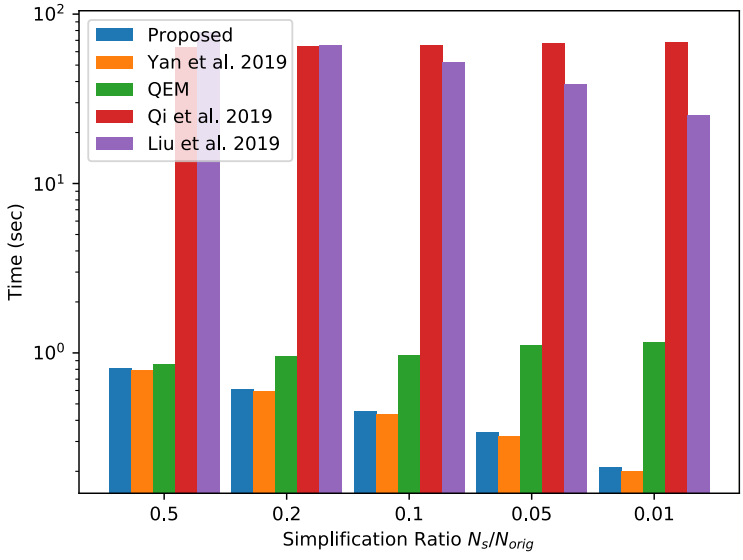


Figure 5.7: Average time of simplification for the proposed and the baseline methods.

5.5.6 Classification of simplified point clouds

To further assess the simplification quality of the simplified point clouds, a pretrained shape classification model was utilized, and its classification accuracy on the simplified point clouds was measured. This approach allowed for the evaluation of the preservation of high-level semantics using an external objective judge, such as a neural network. Specifically, a PointNet [20] model was trained on the train split of the TOSCA dataset. The proposed method and the baseline methods were used to simplify the remaining test split. Figure 5.8 presents the classification performance, in terms of accuracy, of the compared methods at different simplification ratios. It is important to note that the scope of this experiment is to demonstrate that the simplified point clouds produced by the proposed method can be better identified, by a pretrained classifier, compared to the ones produced by the baselines. Results of the original test set performance were reported at simplification ratio equal to 1. It can be easily observed that the proposed model degrades with a smaller slope at extreme simplification ratios, compared to the baseline models. This strengthen the argument that the proposed method retains the salient features that characterise each object and indicate its identity. Arguably, the performance drop of the baseline models could be attributed to the uniform way of sampling points that may drive to decimation of salient points that characterize the point cloud. In contrast, perceptual preservation guided the perceptually guided simplification of the proposed method selects points according to their visual importance, aiming to secure that the salient features will be decimated last.

5.5.7 Simplification under noise conditions

To both quantitatively and qualitatively evaluate the performance of the proposed method in the presence of noise the pretrained model was fed with point clouds distorted with Gaussian noise ($\sigma = 1$). Quantitative experimental result are summarized in Table 5.5, with the proposed method to exhibit the best performance for almost all metrics (NC,RE,SDM) across all simplification ratios, and even the best second for structure preservation (CD). Such findings reveal the anti-noising capabilities of latent space sampling that, compared to xyz-space sampling, is less affected by the outlier noisy points. As can be observed by inspecting Figure 6.9, the proposed method preserves most of the structural characteristics of the input, without being affected from the outlier noisy points as much as the baseline methods. In contrast, sampling points directly from the xyz-space using the FPS method produces noisy outputs following the noisy patterns of the inputs. Similarly, QEM selects noisy points in order to minimize the quadric error of the input

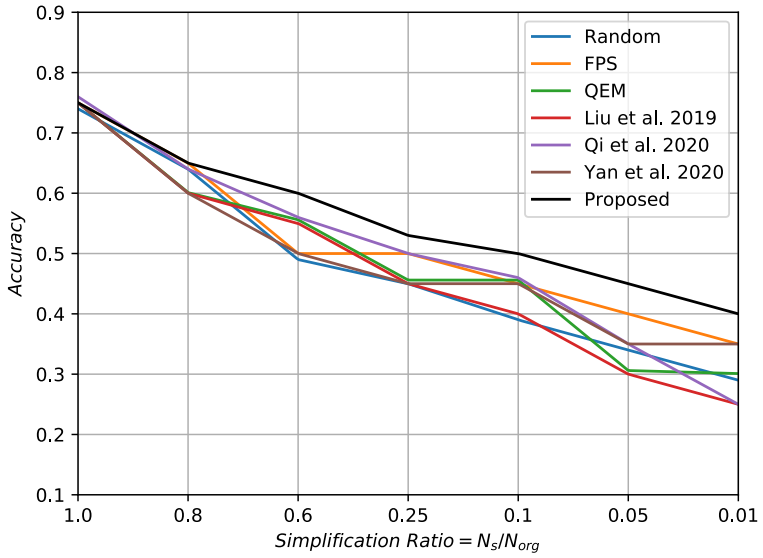


Figure 5.8: Classification accuracy of the pretrained PointNet++ on simplified point clouds at different ratios.

planes, such as the outliers in cat's foot and human hand (shown in zoomed areas).

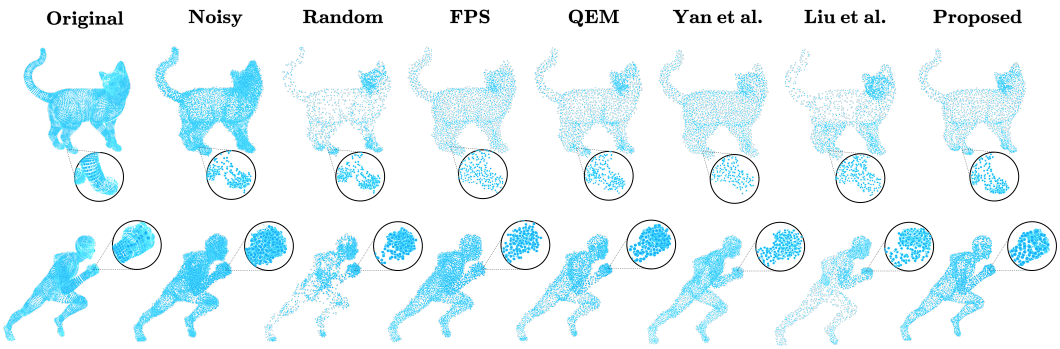


Figure 5.9: Qualitative comparison between the baseline and the proposed methods for point clouds with Gaussian noise addition.

Table 5.5: Simplification performance tested on TOSCA dataset with addition of Gaussian noise. Best approaches are highlighted in **bold** and second best in **red**. "Proposed w/o noise" is reported for reference.

	N_s/N_{org} = 0.2				N_s/N_{org} = 0.1				N_s/N_{org} = 0.05			
Method	CD	NC	RE($\times 10^{-4}$)	SDM($\times 10^{-4}$)	CD	NC	RE($\times 10^{-4}$)	SDM($\times 10^{-4}$)	CD	NC	RE($\times 10^{-4}$)	SDM($\times 10^{-4}$)
Random	2.71	0.37	6.56	9.20	4.43	0.38	6.74	14.39	7.78	0.39	6.85	23.63
TCP	24.7	0.48	6.30	9.27	37.2	0.49	6.58	14.83	53.5	0.51	**6.77**	23.45
FPS	2.74	0.34	**6.25**	9.89	4.28	**0.36**	**6.34**	15.86	6.83	0.39	6.81	25.03
QEM	**1.92**	**0.31**	6.61	**9.14**	**2.57**	0.36	6.81	**14.53**	**3.85**	**0.37**	6.93	**23.12**
Liu *et al.* [68]	3.04	0.35	7.14	9.48	3.99	0.39	7.36	16.21	6.74	0.412	7.61	26.35
Qi *et al.* [175]	3.21	0.33	7.22	11.14	4.31	0.35	7.54	18.34	7.13	0.39	7.81	27.52
Yan *et al.* [183]	2.95	0.35	6.54	10.04	4.30	0.39	7.30	16.01	7.11	0.41	7.12	25.14
Proposed	**2.50**	**0.33**	**6.13**	**8.81**	**3.96**	**0.35**	**6.24**	**14.42**	**6.46**	**0.37**	**6.34**	**22.31**
Proposed w/o noise	1.12	0.29	3.91	5.01	2.21	0.31	4.41	7.84	4.93	0.33	4.93	16.54

5.5.8 Simplification of Real-World Point Clouds

A significant application of point cloud simplification methods is to sub-sample points of real-world scanners that generate million of points from the representative surface. To test the performance of the proposed method on such scenario, the Torronto3D dataset [209] was utilized, which contains outdoor point clouds acquired with LIDAR sensors. Again, the pretrained model on TOSCA dataset was used, without further training or tuning. Quantitative results summarized in Table 5.6 demonstrate that the proposed method outperforms baseline methods in perceptual quality measures (CE, RE, SDM).

Table 5.6: Simplification performance tested on outdoor point cloud from Torronto3D dataset. Best approaches highlighted are highlighted in **bold**. The proposed method model is trained with TOSCA dataset.

	Torronto3D														
	N_s/N_{org} = 0.2					N_s/N_{org} = 0.1					N_s/N_{org} = 0.05				
Method	CD	NC	CE($\times 10^{-2}$)	RE($\times 10^{-4}$)	SDM($\times 10^{-4}$)	CD	NC	CE($\times 10^{-2}$)	RE($\times 10^{-4}$)	SDM($\times 10^{-4}$)	CD	NC	CE($\times 10^{-2}$)	RE($\times 10^{-4}$)	SDM($\times 10^{-4}$)
Random	0.31	0.577	8.15	11.27	0.47	0.62	0.634	8.89	11.56	0.48	1.27	0.679	9.15	11.91	0.48
TCP	5.90	0.894	15.64	14.47	0.58	7.40	0.912	12.41	12.95	0.53	9.58	0.912	13.02	12.61	0.53
FPS	**0.17**	**0.509**	6.42	11.22	0.46	**0.34**	**0.565**	7.01	11.32	0.46	**0.70**	**0.619**	7.51	11.37	0.47
Proposed	0.18	0.512	**5.67**	**11.02**	**0.34**	0.37	0.595	**6.35**	**11.10**	**0.38**	0.75	0.644	**6.88**	**11.15**	**0.41**

Although FPS achieves the lower CD and NC errors and produces smooth results that minimize the overall shape loss, it fails to preserve essential details of the object. Figure 5.10 shows examples of the simplified lidar point clouds at different simplification ratios generated by the proposed method.

5.6 Implementation Details

The projector network was implemented using three multi-layer perceptrons (MLP) followed by Batch Normalization [210] and ReLU activation functions [166]. The filter sizes were set to 64. The GNN following the stacked MLPs was also ReLU activated with a

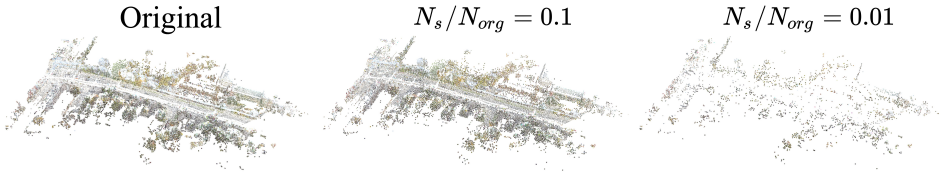


Figure 5.10: Simplification of real-world scans using the proposed method. Figure better viewed in zoom.

filter size of 64. A relatively small neighborhood size of k=7 nearest neighbors was used as the input graph connectivity, following [26, 24]. The initial point of FPS was randomly selected, as it did not show any influence on the performance. For each cluster center selected by FPS, 15 neighbors were selected, as suggested in the literature. The filter size of the attention-based refinement layer was set to 3, mapping the (64+3) features of the selected points to (x, y, z) coordinates. The proposed model was trained for 150 epochs with a learning rate of 0.001 and a weight decay of 0.99 on every epoch using the Adam optimizer [167].

5.7 Conclusion

The work in this Chapter emphasizes the proposal of a learnable, neural-based simplification approach to overcome the inefficiencies of traditional greedy simplification methods. A learnable point cloud simplification method is presented that aims to preserve salient features while retaining the global structural appearance of the input 3D object. Using three learnable modules, large-scale point clouds can be simplified in real-time, addressing the limitations in the literature regarding computational complexity. In an extensive series of both quantitative and qualitative experiments, the proposed method not only outperforms its counterparts under most perceptual criteria but also exhibits zero-shot capabilities. In the next Chapter, we will attempt to adapt the proposed method to mesh structures using a more sophisticated triangulation process. Instead of relying on off-the-shelf triangulation algorithms on top of the point cloud simplification model, the aim is to extend the proposed method to predict the triangulation of the simplified model by utilizing the priors of the input mesh.

CHAPTER 6

3D MESH SIMPLIFICATION

Contents

DESPITE the advent in rendering, editing and preprocessing methods of 3D meshes, their real-time execution remains still infeasible for large-scale meshes. To ease and accelerate such processes, mesh simplification methods have been introduced with the aim to reduce the mesh resolution while preserving its appearance. In this Chapter, the novel task of learnable and differentiable mesh simplification is tackled. Compared to traditional simplification approaches that employ a greedy iterative process of collapsing edges, a fast and scalable method is proposed that simplifies a given mesh in one-pass. The proposed method unfolds in three steps. Initially, a subset of the input vertices is sampled using a sophisticated extension of random sampling. Then, a sparse attention network is trained to propose candidate triangles based on the edge connectivity of the sampled vertices. Finally, a classification network estimates the probability of a candidate triangle to be included in the final mesh. The fast, lightweight, and differentiable properties of the proposed method make it possible to be plugged into any learnable pipeline without introducing a significant overhead. The sampled vertices and the generated triangles are evaluated under several appearance error measures, and their performance is compared against several state-of-the-art baselines. Furthermore, it is showcased that the running performance can be up to 10× faster than traditional methods.

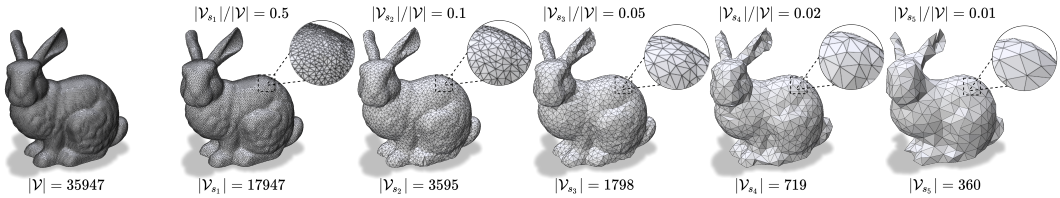


Figure 6.1: The proposed fast and learnable method for mesh simplification that generates simplified meshes in real-time.

6.1 Introduction

Triangle meshes remain the most popular 3D structure to represent a surface. The advent of 3D scanning devices have made feasible to collect highly detailed 3D meshes that typically hold thousand of faces. However, extreme details lead to enormous memory requirements that limit their usage. A wide range of applications including rendering and editing along with their mobile implementations, require lightweight meshes in order to achieve real-time processing. Additionally, many monocular 3D reconstruction methods that utilize analysis-by-synthesis are required to be computational efficient and differentiable in terms of their topologies. Such types of iterative optimization methods can drastically benefit from an differentiable on-the-fly simplification technique that reduce their computational footprint.

Mesh simplification is a long studied problem, with an immense amount of methods developed to sustainably reduce the size of the original mesh without extremely distorting its appearance. Traditional simplification techniques decimate the input mesh in a greedy-fashion by prioritizing vertices and edges according to a cost function [53, 58]. However, in large-scale objects scenario, simplifying over 90% of the original mesh size requires iterating through thousands of vertices resulting in an inevitable computational burden. In addition to their computational footprint, traditional simplification techniques are non-differentiable and thus can not be used directly in end-to-end training processes that optimize the mesh surface. To alleviate the aforementioned limitations, a learnable strategy for mesh simplification was proposed that reduces both time and computational requirements and provides a plug-and-play method, ready to be adapted in any differentiable framework.

A major barrier that limits learnable simplification methods is the discrete nature of the mesh connectivity, i.e. edges and triangles. Although mesh simplification can be achieved in a two-step process using a learnable sampling method followed by an off-the-shelf triangulation algorithm(e.g. Delaunay or Ball Pivoting) as show in Chapter 5, such setting,

apart from being time consuming, limits the direct optimization of the mesh surface. In particular, in the previous Chapter 5, a learnable point cloud simplification technique was proposed, that is able to preserve the curvature features of the input point clouds. This method can be also extended to mesh simplification task by utilizing off-the-self algorithms to triangulate the resulting simplified point cloud. Nevertheless, such methodology requires careful selection of the triangulation parameters for every sample. Importantly, the original topology of the mesh is neglected, i.e. the mesh connectivity might be totally different, and cannot be directly optimized in a learnable process. Recently, several approaches have been proposed to solve the non-trival task of differential triangulation using soft relaxations of the discrete setting. However, most of them are considered impractical since they are applied to volumetric representations [211], demand iterative optimizations for the triangulation of each mesh [212] or require 2D parametrizations [213, 214]. The aim of this work is to devise a simple but intuitive differentiable process to directly triangulate the 3D points in one-pass. To do so, the triangulation process was modeled by generating a distribution over possible edges and triangles and select the ones that preserve the appearance of the original mesh.

In this Chapter, the first learnable mesh simplification method is proposed, that generates both points and triangulation of the simplified mesh. In contrast to previous Chapter 5, a soft relaxation of the discrete triangulation setting is proposed by learning the mesh connectivity distribution in an unsupervised manner. The proposed method can simplify meshes of any scale in real-time by using an extremely efficient point sampling method and a lightweight triangle classifier. To follow the initial mesh appearance, the distribution of the edges was constrained to the priors defined by the original mesh connectivity. The proposed method is fully-differentiable and can be adapted to any training procedure without a significant computational footprint. Finally, the proposed method can generalize to out-of-distribution samples exhibiting zero-shot capabilities.

The main contributions of this Chapter are summarized as:

- The first learnable mesh simplification framework is proposed that is trained to both select vertices and generate the underlining triangulation of the surface.
- The proposed model is fully differentiable and can be directly adapted to any learnable framework.
- An efficient point selection method is introduced that extends the naive random sampling [215] to a trainable module that samples vertices from the underlying multinomial distribution.

- Finally, a simple but intuitive triangulation strategy is proposed, which can be adapted to point cloud meshing.

6.2 Related Work

Learnable Triangulation

Albeit surface reconstruction method have been extensively studied over the years, less attention has been devoted to learnable and differentiable triangulation methods. Most of the existing techniques in the literature generate mesh surfaces by estimating implicit funtions [216, 217], calculating voxel grids and occupancy fields [218, 219] or deforming a template mesh [201]. Less progress has been established to the challenging task of direct point set triangulation, mainly due to the discrete nature of the edges that compose the mesh connectivity. PointTriNet [212] utilizes two models to suggest and classify triangles in local patches, enforcing the selection of watertight and manifold triangles using ad-hoc losses. A similar principle is also used in [220] where the candidate triangles are iteratively filtered using the estimated ratio of the geodesic versus the euclidean distance. Recently, there has been an exertion to employ traditional Delaunay surface triangulation into the learnable triangulation process. In [213], the authors propose to learn a parametrization that maps the input point patches to two-dimensional spaces and triangulate them using Delaunay technique. In a similar manner, a soft relaxation of weighted Delaunay triangulation was also proposed [214] that enables gradient flow, using an inclusion score to discard proposed triangles from the final mesh. Similarly to [213], the input point cloud is triangulated to a spectral partitioned 2D subspace produced using Least-Squares Conformal Map parametrization. In this Chapter a modular architecture is proposed that directly learns to generate a triangulated version of the input vertex set without adhering to any kind of 2D projection and mapping.

6.3 Method

The architecture of the proposed model is composed by mainly three components: the *Point Sampler*, the *Edge Predictor* and the *Face Classifier*.

All modules are fully differentiable and are trained in an end-to-end fashion. Figure 6.2 illustrates an overview of the proposed method.

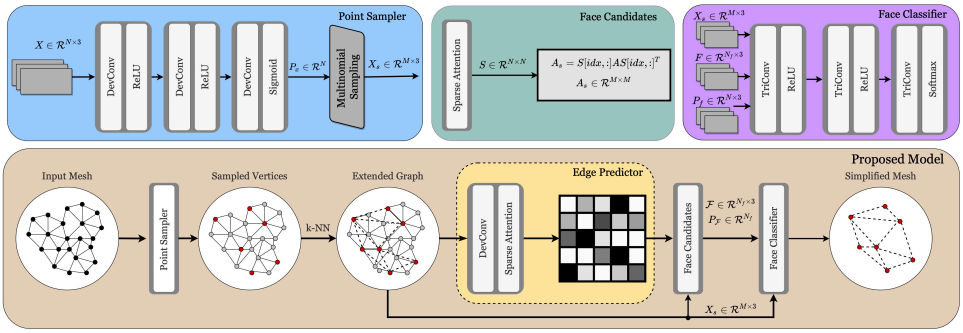


Figure 6.2: Overview of the proposed model (bottom block). Initially, Point Sampler module (top left) samples a subset of the input vertices from the generated multinomial distribution. To avoid isolated nodes, a k-NN graph is constructed to extend the already existing edges of the graph. Following that, a sparse attention layer weights the connectivity between nearby vertices and generates a set of candidate triangles (top middle). Finally, Face Classifier module defines a graph over the proposed faces and assigns a probability to each triangle based on their relative features (top right).

6.3.1 Point Sampler

The first module of the proposed model is a network that samples the vertex set in a way that the mesh structure will remain intact. Previous works, utilized Farthest Point Sampling (FPS) to sample input point sets since it has been shown that it accurately preserves the underlying shape of the object [22]. However, the iterative nature of FPS is not scalable making it impractical for large scale point sets. In contrast, as shown in [215], random sampling is extremely fast, size agnostic, holding an $O(1)$ complexity opposed to the FPS's quadratic $O(N^2)$ complexity. Based on these observations, an extension to random sampling is proposed that utilizes a sophisticated learnable module that breaks the uniform hypothesis of random sampling and samples nodes under the assumption of a multinomial distribution. To do so, the Point Sample module was trained to assign an inclusion score to each point in the vertex set which enables fast sampling.

Given that the simplified point set should approximate the structure of the input point set, the Point Sampler module needs to be trained to select points that provide the best coverage of the input space. Taking into account that nearby points will also have similar latent descriptors, a graph neural network (GNN) approach was used, that will intuitively provide shape insights for the vertices. An update rule based on the relative vertex coordinates was employed that can approximate the deviation of a point from its neighborhood

as:

$$\mathbf{f}_i = \sigma\left(\mathbf{W}_\phi \max_{j \in \mathcal{N}(\mathbf{x}_i)} \mathbf{W}_\theta(\mathbf{x}_i - \mathbf{x}_j)\right) \tag{6.1}$$

where $\mathcal{N}(\mathbf{x}_i)$ denotes the neighbouring points of $\mathbf{x}_i$, $\mathbf{W}_\phi, \mathbf{W}_\theta$ are learnable parameters and $\sigma(\cdot)$ a non-linearity. This GNN layer will be referred as **DevConv**.

With such formulation, the advantage is two-fold. Primarily, the point descriptors may better describe the topological characteristics of a given point, where points in sharp and rough regions will receive larger values compared to smooth areas. Secondly, the sampling module gains robustness to noise, since the combination of the *maximum* as aggregation function and the relative coordinates provides to the network the ability to easily identify outliers.

6.3.2 Edge Predictor

Following the Point Sampler, the Edge Predictor module is responsible to predict the connectivity between the sampled points. To do so, an initial extension of the original mesh connectivity is performed by inserting edges defined by a k-nn graph over the sampled points to avoid isolated nodes in the final mesh. The extended graph $\mathcal{G}_{ext}$ is then processed by a DevConv layer followed by a sparse self-attention layer [11] that predicts the probability that the point $\mathbf{x}_i$ is connected with the point $\mathbf{x}_j$. Such probability is formulated as:

$$\mathbf{S}[i,j] = \frac{\exp\left(\left(\mathbf{W}_q\mathbf{f}_j\right)^T\left(\mathbf{W}_k\mathbf{f}_i\right)\right)}{\sum\limits_{k \in \mathcal{N}(\mathbf{x}_i)} \exp\left(\left(\mathbf{W}_q\mathbf{f}_j\right)^T\left(\mathbf{W}_k\mathbf{f}_k\right)\right)} \tag{6.2}$$

where $\mathbf{W}_q, \mathbf{W}_k$ are learnable parameters and $\mathbf{f}_i$, $\mathbf{f}_j$ are the features of points $\mathbf{x}_i, \mathbf{x}_j$.

To avoid having edges of equal probability between nearby points, DevConv was utilized to enable feature dissimilarity between them. Finally, the estimated adjacency matrix is defined using the product of the estimated probabilities and the original adjacency matrix, following the formulation of [46], as:

$$\hat{\mathbf{S}} = \mathbf{S}[:, \texttt{idx}] \in \mathbb{R}^{N \times M} \tag{6.3}$$

$$\mathbf{A}_s = \hat{\mathbf{S}}^T \cdot \mathbf{A} \cdot \hat{\mathbf{S}}, \quad \mathbf{A}_s \in \mathbb{R}^{M \times M} \tag{6.4}$$

where `idx` refers to the vertices selected by the point sampler.

The motivation behind the use of the product between the estimated and the original

connectivity is to enforce the edges of the simplified mesh to respect the original topology. Note that all of the aforementioned multiplication operations are between the sparse matrices and can be calculated very efficiently.

The set of candidate faces can be easily constructed from the non-zero entities of the simplified adjacency matrix $\mathbf{A_s}$. In particular, if $\mathbf{A}_s[i,j], \mathbf{A}_s[i,k], \mathbf{A}_s[j,k]$ have all non zero values, a candidate triangle $t = (i,j,j)$ is constructed. The initial inclusion probability p_t^{init} is assigned to triangle t is defined as:

$$p_t^{init} = \frac{1}{3}(\mathbf{A}_s[i,j] + \mathbf{A}_s[i,k] + \mathbf{A}_s[j,k]), \quad i,j,k \in t \tag{6.5}$$

, where i,j,k are the vertices of triangle t.

This initial face inclusion probability can be thought as a prior that will be invoked by the Face Classifier to produce the final (posterior) probability of the edge.

6.3.3 Face Classifier

The face classifier is responsible to assign to each triangle an inclusion score p_t that captures the probability of a triangle to be present in the simplified mesh. To estimate this probability, a k-nn graph $\mathcal{G}_{tri}$ is initially constructed, which connects each candidate triangle with its k-neighbours based on their barycenter distances. Then a GNN module, namely **TriConv**, that acts on $\mathcal{G}_{tri}$ embeds faces to the latent space. To better capture the interactions of two triangles n and m in space, a relative position encoding $\mathbf{r}_{n,m}$ was utilized which can be defined as:

$$\begin{gathered}\mathbf{r}_{n,m} = [(\mathbf{t}_n^{min} - \mathbf{t}_m^{min}) || (\mathbf{t}_n^{max} - \mathbf{t}_m^{max}) || (\mathbf{b}_n - \mathbf{b}_m)], \\ \mathbf{t}_n^{max} = max(\mathbf{e}_{ij}^n, \mathbf{e}_{ik}^n, \mathbf{e}_{jk}^n)\end{gathered} \tag{6.6}$$

where $\mathbf{t}_n^{max}, \mathbf{t}_n^{min} \in \mathcal{R}^3$, $\mathbf{e}_{ij}^n$ the edge vectors $\mathbf{x}_i - \mathbf{x}_j$ for triangle n; b_n, b_m the barycenters of triangles n, m and $||$ the concatenation operation. Finally, the update rule of TriConv can be defined as follows:

$$\mathbf{f}_n^{(l)} = \sum_{k \in \mathcal{N}(t_n)} MLP([\mathbf{r}_{n,k} || (\mathbf{f}_n^{(l-1)} - \mathbf{f}_k^{(l-1)})]) \tag{6.7}$$

where $\mathcal{N}(t_n)$ the neighborhood and $\mathbf{f}_n^{(l-1)}$ the previous feature of face n. In order to

generate the final inclusion probability p_t three TriConv layers were stacked, toped with a softmax activation function. The prior probability p_t^{init} was used as the initial feature of each triangle.

6.3.4 Losses

To train the proposed framework a set of loss functions was used in order to that oblige the simplified meshes to preserve the visual appearance of the originals. The basic idea underlying the utilized losses is to enforce the selection of salient points that are representative of the mesh structure and to penalize badly formed triangles.

Probabilistic Chamfer Distance: To improve uniform sampling, the Point Selector is designed to assign high probabilities to the points that cover the surface of the point cloud. This is achieved by sampling a sufficient number of points from the input mesh surface $\mathcal{S}$ and measuring the distance between the points sampled from the Point Sampler and the points sampled from the input surface. Mathematically, this is formulated using a modified probabilistic Chamfer distance:

$$d_{\hat{\mathcal{P}},\mathcal{P}_s} = \sum_{\mathbf{y}\in\mathcal{P}_s} p(\mathbf{y}) \min_{\mathbf{x}\in\hat{\mathcal{P}}} \|\mathbf{x}-\mathbf{y}\|^2 + \sum_{\mathbf{x}\in\hat{\mathcal{P}}} p(\mathbf{y}) \min_{\mathbf{y}\in\mathcal{P}_s} \|\mathbf{x}-\mathbf{y}\|^2 \tag{6.8}$$

where $\hat{\mathcal{P}}$ denotes the points sampled from the input surface $\mathcal{S}$, $\mathcal{P}_s$ the sampled points and $p(\mathbf{y})$ their respective probabilities. Given that this loss is differentiable only with respect to the probabilities of the sample points it is only applied to the Point Sampler module. This means that one can pre-train the Point Sampler and adapt it to any learnable framework.

Probabilistic Surfaces Distance: To avoid having triangles in regions that are not existing in the original mesh and penalize the presence of surface holes, a Chamfer-inspired loss was designed, which measures the distance between the ground truth and the probabilistic surface. In this setting, the forward term of the distance, i.e. the distance between the generated surface $\mathcal{S}_s$ and the original, enforces triangles to fit the ground truth surface $\mathcal{S}$. This term can be modeled by using the barycenters of the two surfaces as follows:

$$d^f_{\mathcal{S},\mathcal{S}_s} = \sum_{\hat{\mathbf{b}}\in\mathcal{S}_s} p_{\hat{\mathbf{b}}} \min_{\mathbf{b}\in\mathcal{P}} \|\hat{\mathbf{b}}-\mathbf{b}\|^2 \tag{6.9}$$

where $\mathbf{b}$ stands for the barycenters of the ground truth surface triangles, $\hat{\mathbf{b}}$ the barycenters

of the generated triangles. Each barycenter holds a probability $p_{\hat{\mathbf{b}}}$ that it is equal to its corresponding triangle $\hat{t}$ probability $p_{\hat{\mathbf{t}}}$. In this manner, barycenters of triangles in non existing regions, e.g. a triangle connecting the dogs legs, will have greater distance compared to barycenters of triangles in existing regions of the ground truth surface.

In contrary to the forward term, the reverse term of the distance function aims to penalize areas with small probabilities, i.e. areas that when discarded may result into the introduction of surface holes. Mathematically this can be defined as follows:

$$d^r_{\mathcal{S},\mathcal{S}_s} = \sum_{y \in \mathcal{S}_s} p_{\mathbf{y}} \min_{\mathbf{x} \in \mathcal{P}} \|\mathbf{x} - \mathbf{y}\|^2 + \left(1 - p_{\mathbf{y}}\right) \frac{1}{k} \sum_{k} p_{t_k} \|\mathbf{x}_{t_k} - \mathbf{y}\|^2 \tag{6.10}$$

where $\mathbf{x}$ denotes a point from the ground truth surface $\mathcal{S}$ and $\mathbf{y}$ a point from the generated surface $\mathcal{S}_s$ with probability $p_{\mathbf{y}}$. The second term of equation (6.10) estimates the average distance between point $\mathbf{y}$ and its k-nearest triangles t_k in the generated surface $\mathcal{S}_s$ apart from the one that point $\mathbf{y}$ belongs to. This last term can be intuitively conceptualized as the error introduced when the triangle in which $\mathbf{y}$ belongs is not present in the generated triangulation. To make the reverse term robust, a sufficient amount of points were sampled from each generated triangle.

Triangle Collision: To avoid having triangles that penetrate each other, a loss term was introduced that directly penalizes the probabilities of such triangles. The collision of a triangle is measured in terms of line segments (i.e. edges of nearby triangles) penetrating its surface. In particular, the planes defined by each face are computed, and nearby triangles formed from edges that penetrate these planes are penalized. The penalty applied to such irregular triangle is proportional to the number of planes that it penetrates and it is defined as:

$$\mathcal{L}_c = \frac{1}{|\mathcal{F}_s|} \sum_{t \in \mathcal{F}_s} p_t m_c(t) \tag{6.11}$$

where p_t denotes the probability of triangle t, $m_c(t)$ the number of faces penetrated by triangle t and $\mathcal{F}_s$ the set of generated triangles.

Although triangle collision loss may be sufficient to penalize triangles that penetrate the surface of their neighboring triangles, it can not capture and penalize overlapping triangles with parallel planes. To address this limitation, two additional losses were introduced that penalize such triangles, namely the edge crossings loss $\mathcal{L}_e$ and the overlap loss $\mathcal{L}_o$.

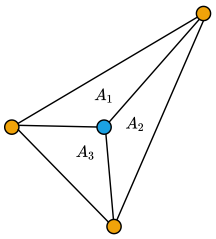
Edge Crossings: Edge crossings, in terms of line segments (edges), of nearby triangles are calculated, and triangles that carry an edge crossing another edge are directly penalized. The edge crossings loss between two triangles was calculated by splitting each triangle into the three lines that define their edges. The points of intersection between all nine possible edge-line combinations of two neighboring triangles are then calculated. The points of intersection are validated to belong inside the line segments (i.e., edges) of the query triangle. The frequency at which a triangle penetrates the edges of its neighboring triangle is measured, and it is penalized using the formula below:

$$\mathcal{L}_e = \frac{1}{|\mathcal{F}_s|} \sum_{t \in \mathcal{F}_s} p_t m_e(t) \tag{6.12}$$

where p_t denotes the probability of triangle t, $m_e(t)$ the number of edges that triangle t crosses and $\mathcal{F}_s$ the set of generated triangles. Similar to collision loss, the penalty applied to each triangle is proportional to the number of faces that it penetrates.

Overlapping Triangles: Finally, to avoid overlapping triangles in space, a sufficient number of points were sampled from each generated face and compute an estimate that belongs to a given face. This can be efficiently implemented by measuring the sum of the areas produced by the sampled point and the vertices of each of the k-nearest triangles. In particular, 100 points were sampled from each triangle and calculate their distances from the 50-nearest triangles. The aim is to enforce the sampled point to belong to only one triangle.

To identify if a point belongs to a given triangle, the three areas A_1, A_2, A_3 produced by substituting each triangle's vertex with the query point were calculated and if the sum of the three produced areas equals the area of the triangle was crosschecked. Additional provision was applied for triangles that share parallel planes by slightly increasing the distance tolerance to the axis that is vertical to the triangle's plane. Finally, a penalty was applied to all triangles that share a sampled point. Similar to the Edge Crossing loss, the penalty applied to the triangle is proportional to the number of triangles that overlaps:



$$\mathcal{L}_o = \frac{1}{|\mathcal{F}_s|} \sum_{t \in \mathcal{F}_s} p_t m_o(t) \tag{6.13}$$

where p_t denotes the probability of triangle t, $m_o(t)$ the number of triangles that triangle t overlaps.

Overall Objective Finally, the overall loss to be minimized is formed as:

$$\mathcal{L} = d_{\mathcal{P}_1,\mathcal{P}_2} + d^f_{\mathcal{S}_1,\mathcal{S}_2} + d^r_{\mathcal{S}_1,\mathcal{S}_2} + \lambda_c \mathcal{L}_c + \lambda_e \mathcal{L}_e + \lambda_o \mathcal{L}_o \tag{6.14}$$

where $\lambda_c, \lambda_e, \lambda_o$ are hyperparameters that scale the loss functions.

6.4 Implementation Details

Point Sampler: The Point Sampler is constructed using three stacked DevConv layers, followed by a ReLU activation. A hidden dimension of 64 is selected to ensure a lightweight and fast model. Deeper architectures and larger latent spaces were experimented with, but no significant increase in performance was observed compared to the trade-off in run-time.

Edge Predictor: The nodes are passed to the edge predictor after extending the original adjacency matrix by adding links between the nodes sampled by the Point Sampler. This is done by constructing a k-nn graph for the sampled node set, with $k = 15$. The extended graph is then passed through a DevConv layer with 64 hidden dimensions. DevConv is utilized to allow the model to assign sparse attention weights. The use of Linear or vanilla-gnn layers would result in high attention scores for points with similar features, which may not always be useful. DevConv enables larger variations between point features, leading to more sparse attention weights.

Face Classifier: Three stacked TriConv layers were utilized, followed with ReLU activation to encode triangle to the latent space, formed by its k-nearest neighbors. The number of the triangle neighbors was set to k=20, selected by their respective distances to their barycenters. The triangle features are encoded to an 128-dimensional embedding space with additional 9 features from the triangles relative coordinates. The first TriConv layer takes as input the initial triangle probability predicted by the edge predictor. The final TriConv layer is topped with a softmax layer to constrain the triangle probabilities to the $(0, 1)$ interval.

Training: The model was trained for 150 epochs with learning rate of $1e-5$ and a weight decay of 0.99 on every epoch using the Adam optimizer [167]. The generated simplified meshes are produced by selecting only the faces with a probability above 0.5. The generated meshes were further constrained to be manifold by selecting for each edge the two incident to it faces with the highest probability.

6.5 Experiments

In this section, the simplified meshes produced by the proposed framework are evaluated. The effect of the Point Sampler is initially examined, followed by an assessment of the quality of the produced triangulation and the respective run times.

Dataset: To train the proposed method, the benchmark TOSCA dataset [205] was utilized, consisting of 80 high-resolution meshes. The same train-test split as in [10] was used, testing the model on topologies not present in the training set. This allows the devised model to be directly applied to out-of-distribution meshes and generalize across different topologies.

Baselines: Several baselines were selected for comparison, each with different properties. The popular quadric mesh simplification (QEM) [58] was chosen as a baseline, known for its efficiency and popularity in mesh simplification. Additionally, two learnable and differentiable triangulation methods, PointTriNet [212] and DSE [214], were selected to triangulate point clouds sampled with FPS, providing an alternative for differentiable mesh simplification. Lastly, the proposed method was compared against a recently introduced learnable point cloud simplification method [10], which utilizes the Ball-Pivoting algorithm to triangulate the simplified point clouds.

6.5.1 Evaluation of the Simplified Meshes

To evaluate the triangulation performance, the percentage of non-watertight edges (WA) was measured, which represents the edges with one, three, or more incident faces. This metric is more suitable for assessing the quality of triangulation when holes are present, as the edges surrounding a hole are still manifold (a watertight edge is manifold, but a manifold edge is not guaranteed to be watertight).

Additionally, to evaluate the simplification performance of each method, $50K$ points were sampled from the surface of each simplified mesh, and the Chamfer distance (CD) and normals error (NE) were measured compared to their corresponding points in the original mesh. The normals error captures the visual appearance of the simplified models, utilizing a normal error that measures the cosine similarity between the normals of the two models. The normals dissimilarity is evaluated using two terms: the forward error (from source to target) and the reverse error (from target to source). For the forward term, the normal differences between each face of the simplified model and its nearest face in the original mesh are measured. The reverse term estimates the closest face of the simplified mesh for

Table 6.1: Quantitative comparison of the proposed and the baseline methods under several simplification ratios. Best approaches are highlighted in **bold** and second best in **blue**. Time is measured in seconds.

	$N_s/N_{org} = 0.05$					$N_s/N_{org} = 0.1$					$N_s/N_{org} = 0.2$					$N_s/N_{org} = 0.5$				
Method	CD	WA	LE	NE	Time	CD	WA	LE	NE	Time	CD	WA	LE	NE	Time	CD	WA	LE	NE	Time
PointTriNet [212]	1.06	12.14	0.98	0.20	107.2	0.47	11.64	0.52	0.17	238.1	0.21	11.48	0.27	0.13	581.9	0.08	14.92	0.12	0.08	1333.3
QEM [58]	**0.45**	**0.00**	**0.94**	**0.14**	**45.6**	**0.22**	**0.00**	0.53	**0.10**	**31.1**	**0.11**	**0.00**	0.27	**0.07**	**28.4**	**0.05**	**0.00**	0.13	**0.03**	**25.6**
DSE [214]	1.39	6.64	0.95	0.23	271.8	0.51	4.38	**0.50**	0.19	490.7	0.24	3.12	**0.26**	0.15	941.0	0.07	**2.39**	**0.12**	0.08	2245.2
Potamias el al. [10]	10.47	8.53	1.23	0.21	105.2	0.83	8.39	0.76	0.17	149.3	0.49	4.72	0.43	0.14	158.7	0.20	4.67	0.22	0.10	183.7
Proposed	**1.02**	**2.17**	**0.90**	**0.19**	**4.1**	**0.42**	**2.21**	**0.47**	**0.15**	**4.2**	**0.19**	**2.49**	**0.24**	**0.11**	**4.2**	**0.06**	3.57	**0.10**	**0.06**	**4.4**

each face of the original mesh and calculates their normal difference. The mathematical formulation of the total normal error is:

$$\mathcal{E}_n = \frac{1}{|\mathcal{P}|} \sum_{\substack{t \in \mathcal{F} \\ q \in NN(t, \mathcal{F}_s)}} \left(1 - \frac{\mathbf{n_t} \cdot \mathbf{n_q}}{\|\mathbf{n_t}\| \|\mathbf{n_q}\|}\right) + \frac{1}{|\mathcal{F}_s|} \sum_{\substack{q \in \mathcal{F}_s \\ t \in NN(q, \mathcal{F})}} \left(1 - \frac{\mathbf{n_t} \cdot \mathbf{n_q}}{\|\mathbf{n_t}\| \|\mathbf{n_q}\|}\right) \tag{6.15}$$

where $\mathbf{n_t}$ denotes the normal of face $t \in \mathcal{F}$ and $NN(t, \mathcal{F}_s)$ the nearest neighbour of face t in simplified mesh $\mathcal{P}_s$.

Finally, to evaluate the preservation of the mesh spectral properties, the Laplacian error was calculated between the original and the simplified mesh, defined as the Mean Squared Error (MSE) over the first 150 eigenvectors of the Laplacian operator of the two meshes. Throughout this Chapter, this spectral error will be referred as Laplacian Error (LE):

$$\mathcal{E}_L = \left(\sum_i \|\hat{\phi}_i - \phi_i\|^2 \right) \tag{6.16}$$

where $\phi_i, \hat{\phi}_i$ denotes the i-th eigenvector of the Laplacian operator for the original and the simplified mesh, respectively.

In Table 6.1 quantitative comparison between the proposed method and the aforementioned baselines is presented. Although the iterative QEM method better preserves the appearance of the simplified mesh, the proposed method achieves smaller Laplacian error while at the same time attain competing performance over all error metrics. On the contrary, the proposed method outperforms all differentiable methods and overcomes the limitations of previous triangulation approaches. In particular, as can be easily observed in Figure 6.3, the proposed method has very few holes compared to PointTriNet [212] as well as less triangles in non-existing regions, such as the triangles occurring between the thigh and the hand (top row Figure 6.3), due to the probabilistic surface distance loss that penalizes the inclusion of such triangles. For further qualitative assessment see also Figure 6.1.

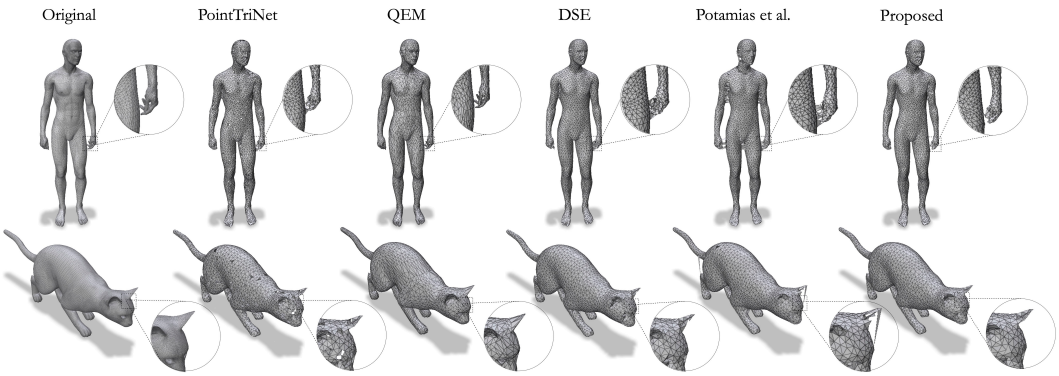


Figure 6.3: Qualitative comparison of the proposed and the baseline methods. The top row contains a human shape simplified by 90% and the bottom row shows a cat model simplified by 80%. Figure better viewed in zoom.

6.5.2 Time Performance

One of the most prominent applications of mesh simplification is inevitably rendering. Real-time rendering requires lightweight model structures, therefore simplification algorithms are commonly equipped in rendering pipelines. To this end, the time performance is of crucial importance for the simplification process. To assess time performance, the execution time was measured for each method to simplify 20 meshes under different simplification ratios. Experimental result presented in Table 6.1 showcase the efficiency of the proposed method, outperforming its baseline counterparts by a large margin. In particular, the proposed method runs up to 10× faster than the optimized QEM method and at least 100× faster than its faster differentiable counterpart. Another important feature of the proposed method is that it remains almost unaffected by the mesh scale, due to the efficient point sampler and the lightweight structure of the face classifier. In summary, the results highlight the fact that the proposed method could be directly plugged into any rendering process without introducing any significant overhead.

6.5.3 Evaluation of Point Sampler

The *Point Sampler* is responsible for the selection of the vertices to be maintained in the simplified mesh. To assess the performance of the sampling module the structural error in terms of i) the Chamfer distance (CD), ii) the details preserved using the two-sided curvature error (Curv) as suggested in [10] and iii) the time required to simplify the input point cloud (in seconds) were measured under several simplification ratios. The proposed

method was compared with FPS [22], uniform random sampling as utilized in [215], and a recently introduced point cloud simplification module [10]. In Figure 6.4 qualitative comparison between the simplified point clouds is illustrated. As can be easily seen, the method proposed in this chapter demonstrates an improved balance between maintaining the overall structure and retaining high-frequency details compared to the curvature-based simplification of the previous Chapter. This will aid the triangulation step that requires performs better under uniform point distributions. Quantitative results are presented in Table 6.2 showcasing that the proposed point sampler outperforms the uniform random sampler as proposed in [215] and also demonstrate a balance between the smooth and the sharp results produced by FPS and the method proposed in Chapter 5 (Potamias *et.al.*[10]). Importantly, the proposed method remains unaffected from the size of the input, achieving a sampling of 4.2 to 24 - times faster than FPS and Potamias *et.al.*[10]. This result clearly demonstrates that the proposed method can be directly utilized to sample large-scale point clouds with the minimal computational overhead, greatly advancing the naive random sampling approach [215]. Finally, the performance of the proposed sampling module was assessed under noisy conditions by training on clean datasets and testing on noisy point clouds by adding zero mean and unit standart deviation Gaussian noise to the data. (right block of Table 6.2). It can be easily observed that the proposed method is less affected by the presence of noise compared to its counterparts by virtue of the DevConv, which encodes points based on the maximum relative features of the neighborhood.

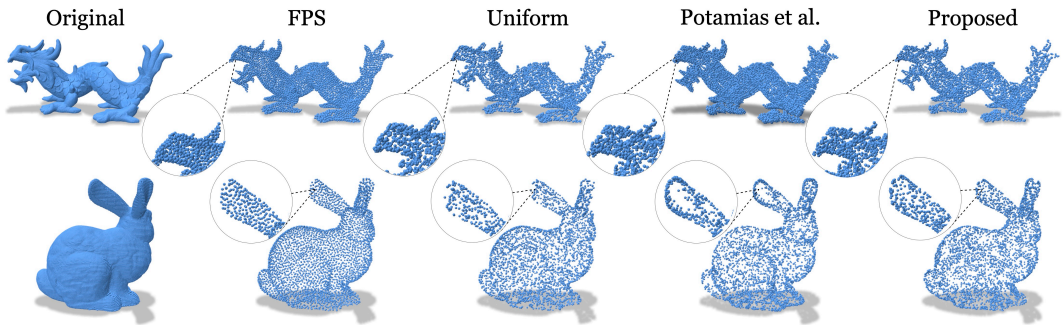


Figure 6.4: Qualitative assessment of the proposed point sampling module. The proposed Point Sampler selects points that preserve both the structure and the details of the input cloud. The proposed method better maintains the structure of the object compared to uniform and [10] counterparts and improves the smooth point clouds produced by FPS module. The top row shows a dragon point cloud simplified to 5% of its input size where as the bottom row shows the bunny shape simplified to 20% of its input size. Please note that both shapes are not present in the TOSCA dataset.

Table 6.2: Quantitative evaluation of the point sampling module at different simplification ratios. The right part of the table includes the simplification performance of the various methods when tested to noisy point clouds.

	w/o Noise									w Noise								
	N_s/N_{org} = 0.8			N_s/N_{org} = 0.2			N_s/N_{org} = 0.05			N_s/N_{org} = 0.8			N_s/N_{org} = 0.2			N_s/N_{org}=0.05		
Method	CD	Curv	Time	CD	Curv	Time	CD	Curv	Time	CD	Curv	Time	CD	Curv	Time	CD	Curv	Time
FPS	**0.01**	2.21	21.2	**0.81**	2.99	7.84	**4.02**	3.42	3.86	2.11	4.07	21.2	3.47	4.35	7.84	7.31	4.79	3.86
Uniform	0.24	2.19	**0.05**	1.85	2.44	**0.04**	6.71	2.73	**0.03**	2.58	3.71	**0.05**	3.67	3.97	**0.04**	7.43	4.20	**0.03**
Potamias et al. [10]	0.03	**2.14**	120.	1.18	2.27	35.1	4.78	2.51	17.7	2.11	3.14	120.	3.38	3.65	35.1	7.61	4.12	17.7
Point Sampler	0.05	2.16	0.05	1.22	**2.24**	0.09	5.12	**2.51**	0.09	**1.99**	**3.12**	0.05	**3.21**	**3.18**	0.09	**7.18**	**4.01**	0.09

6.5.4 Curvature-based Simplification

A significant property of the proposed method is that all of its components are fully differentiable. Thus, it can be seamlessly integrated to an arbitrary iterative framework which requires gradients to flow throughout the optimization process. In this experiment, the capability of the model to be adapted to a trainable pipeline that generates customized simplification was explored. Specifically, the proposed method was fine-tuned to produce simplified meshes that preserve the curvature of the original. To accomplish this, a loss term that quantifies the curvature difference between the original and simplified meshes was incorporated, as proposed in [10]. Experimental results revealed that the fine-tuned method achieved an improvement of 40% (in average) to the curvature error compared to the original version. An example is illustrated in Figure 6.5 where it can be easily observed that the fine-tuned version focuses on preserving the rough details of the original mesh (such as the eyes and the nose-tip) compared to the smooth mesh produced by the untouched version of the proposed method.

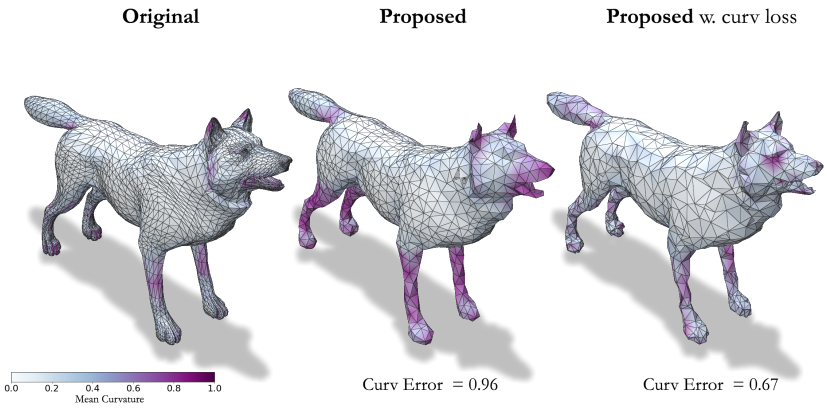


Figure 6.5: Fine-tuning the proposed method for curvature driven simplification.

6.5.5 Evaluation of intrinsic distances

To evaluate the distortion of intrinsic properties in the simplified mesh the geodesic and spectral distances were measured between vertices in a spotted area of the surface. The biharmonic spectral distance is calculated as in [69]. A qualitative color-coded comparison of the cat shape simplified by 90% between the proposed method and the baselines is illustrated in Figure 6.6. All distances are measured form the nose tip of each shape. It can be easily observed that the proposed method preserves both geodesic and spectral distances of the original model compared to the distorted distances produced by the baseline models. In particular, although QEM method manage to satisfactorily preserve the geodesic distances, it introduces enough distortion to the spectral distances that produces a thoroughly different iso-lines compared to the original meshes. Additionally, PointTriNet [212], DSE [214] and Potamias *et.al.*[10] not only they introduce geodesic error, but also their biharmonic distances (spectral) are less smooth compared to the proposed method.

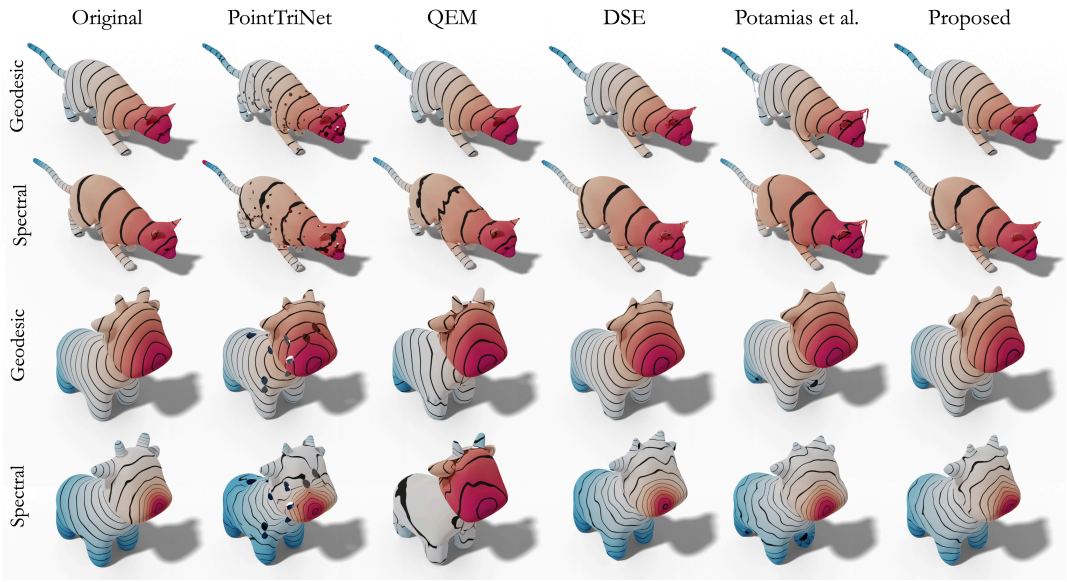


Figure 6.6: Geodesic and Spectral distance comparison between QEM and the proposed method. Distances are measured from the nose-tip of each shape.

6.5.6 Ablation Study

In this subsection, the importance of each loss function holds was assessed by means of an ablation study (see Table 6.3). In particular, one component of the loss function was removed at a time and measure the Chamfer distance (CD), watertightness percetage

Method	$N_s/N_{org} = 0.05$				$N_s/N_{org} = 0.1$				$N_s/N_{org} = 0.2$				$N_s/N_{org} = 0.5$			
	CD	WA	LE	NE	CD	WA	LE	NE	CD	WA	LE	NE	CD	WA	LE	NE
Proposed w/o OV	**1.01**	2.21	0.98	0.20	0.42	2.24	0.52	0.15	**0.19**	2.52	0.27	0.12	**0.06**	3.59	0.12	0.08
Proposed w/o EC	1.02	2.20	0.94	0.20	0.43	2.22	0.53	0.16	0.20	2.50	0.27	0.11	0.06	3.58	0.13	0.07
Proposed w/o TC	1.03	2.22	1.54	0.42	0.42	2.24	0.97	0.36	0.19	2.51	0.78	0.32	0.06	3.58	0.58	0.28
Proposed w/o PSD	2.12	10.24	1.35	0.23	1.33	9.14	0.96	0.19	0.96	6.75	0.77	0.18	0.52	5.97	0.54	0.13
Proposed-Full	1.02	**2.17**	**0.90**	**0.19**	**0.42**	**2.21**	**0.47**	**0.15**	0.19	**2.49**	**0.24**	**0.11**	0.06	3.57	**0.10**	**0.06**

Table 6.3: Quantitative results of the ablation study over the loss functions. OV, EC, TC, PSD denote overlap loss, edge crossing loss, triangle collusion loss and probabilistic surface distance loss, respectively.

(WA), the Laplacian error (LE) and the normals error (NE) at several simplification ratios. It can be easily observed that the models trained without overlap (OV), edge crossing (EC) or triangle collusion (TC) losses, although they achieve similar CD and NE, they fail to preserve the Laplacian of the initial mesh since the irregular triangles perturb the geodesics of the mesh. In addition, as illustrated in Figure 6.7, model trained without the Probabilistic Surface Distance (PSD) loss, generates irregular triangles and introduces holes to the triangulation that increase CD and WA errors. Ablation result for the Probabilistic Chamfer Distance were not reported since it is the only loss applied to the Point Sampler module. All methods were trained on the TOSCA dataset using the same train-test split.

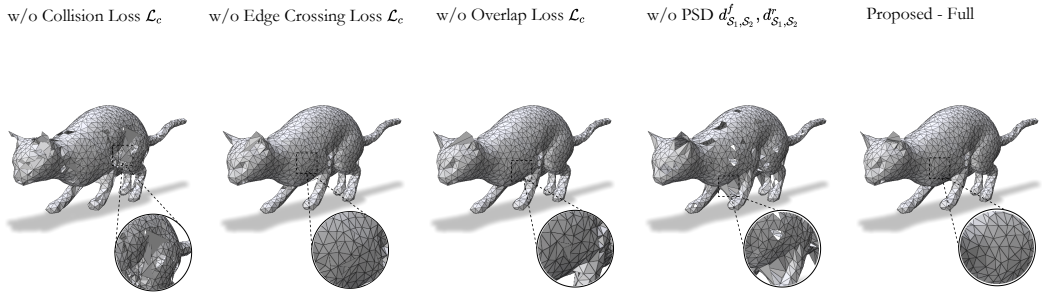


Figure 6.7: Ablation study: Qualitative comparison indicating the contributions of each loss function. Zoomed areas illustrate areas with irregular triangles.

6.5.7 Simplification of Textured Meshes.

Although in this study, the main focus is on mesh shape simplification, it is reasonable to assess the appearance of simplified textured meshes. In this experiment, the texture preservation of simplified meshes was qualitatively examined. A checker pattern was applied to the TOSCA shapes, and the similarity of the simplified models with the originals was evaluated. From Figure 6.8, one can observe that the texture of the QEM method is unsettled and that the sharp corners of the checker have become smoother. In contrast,

the proposed method exhibits significantly less noise and better preserves the details of the original texture. Additionally, the proposed method was assessed in a real-world mesh scenario using a textured mesh from a human face (bottom row Figure 6.8). As observed, the proposed method successfully maintains the texture characteristics of the face.

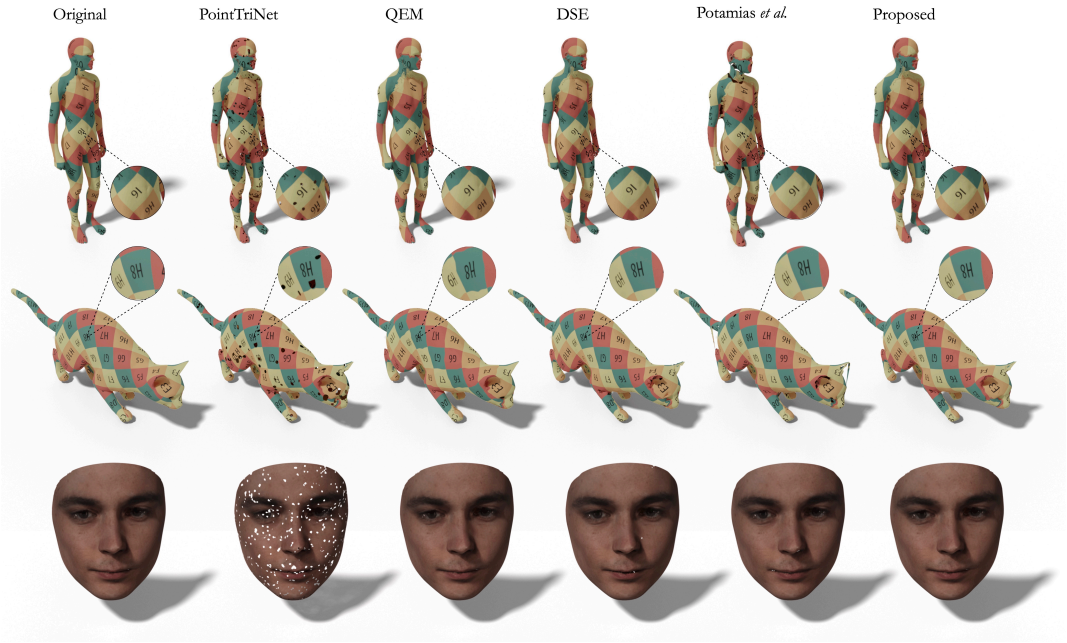


Figure 6.8: Simplification of textured meshes. The human shape model (top row) meshes are simplified by 85%, the cat model (medium row) is simplified by 90% and the face model (bottom row) is simplified by 80%

6.5.8 Simplification of noisy meshes.

Although in real-world applications a noise filtering prepossessing step is always present, the simplification performance of the proposed method under noise conditions was examinded. The devised Point Sampler module is less affected by noise compared to its counterparts due to the DevConv structure. Qualitative comparison between the proposed and the baseline models is illustrated in Figure 6.9. The performance of Point Sampler leads to better triangulation and thus smoother simplified meshes compared to the PointTriNet [212] and DSE [214] modules. QEM method struggles to find the planes associated with each point and generates artifacts to the simplified mesh. Ball pivoting algorithm, utilized in [10], fails to properly triangulate the simplified point cloud and requires careful hyperparameter tuning to avoid irregular triangles. On the contrary, the proposed method

produces smooth results, e.g. the rack of the Centaur model, and manages to generate a simplified model that preserves the appearance of the input.

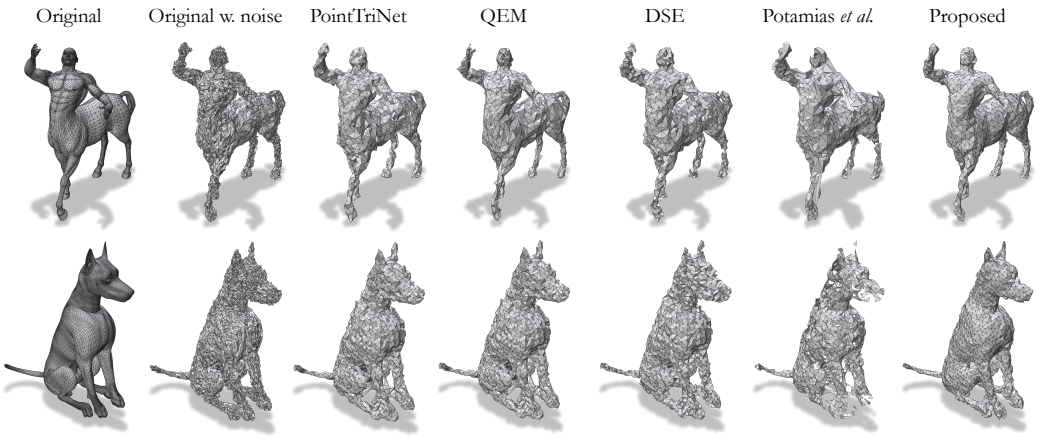


Figure 6.9: Qualitative comparison of the proposed and the baseline methods on noisy mesh simplification. The top row contains a centaur shape simplified by 90% and the bottom row shows a dog model simplified by 90%. Figure better viewed in zoom.

6.6 Conclusion and Limitations

In this Chapter, an attempt to mark a step towards a totally learnable mesh simplification framework was described. The first differentiable mesh simplification model was proposed based on the advances of graph neural networks. The run-time efficiency and lightweight structure of the proposed model enables its direct use in a wide range of differentiable applications. The proposed method outperforms all of its differentiable counterparts. A limitation of the proposed method is that, although the model was enforced to preserve the topology and the manifoldness of the generated meshes using tailor-made loss functions, it can not be explicitly guaranteed. Inevitably, QEM produces smother watertight surfaces with finer details compared to the proposed method. However, QEM comes with a greater computational cost, along with a non-differentiable nature that limits its range of applications. Arguably, the rationale underlying the proposed method and the presented findings of this Chapter will benefit the 3D computer vision community.

CHAPTER 7

CONCLUSION

Contents

TRIANGULAR meshes have been established as the most popular way to represent 3D discretized surfaces. Given that the set of faces that discretize the surface is constructed by a set of ordered vertex triplets, 3D meshes can be considered as undirected graphs embedded on $\mathbb{R}^3$. The limitation of traditional point based processing of 3D sets to adherit local inductive biases to the learning process has emerged the need of exploiting geometric deep learning approaches. The goal of this thesis is to highlight the expressive power of Graph Neural Networks and Geometric Deep Learning to overcome the limitations of traditional 3D shape analysis methods. The thesis can be divided in two parts, the use of mesh convolutions to model 3D shape surfaces and the expressivity of GNNs to identify semantics and simplify 3D shapes. Regarding the first part, mesh convolutions outperformed traditional statistical methods, such as PCA, to model both static as well as dynamic 3D meshes. Additionally, GNNs demonstrated their expressive power to identify salient parts of 3D object and simplify meshes and point clouds while retaining most of their semantics. Finally, in this thesis, GNNs were also explored as message passing networks that propagate information across the mesh faces by creating implicit graph on the mesh face set.

In particular in Chapter 3, the first large scale deformable hand model of the human hand was presented, trained with over than 1200 different subjects, the largest and most enriched dataset so-far. To train a statistical model the scans were brought into correspondence by registering them in a common template. To do so, each hand was rendered from multiple views and 2D landmarks were detected. The landmarks were then lifted to 3D using linear triangulation. Those landmarks are then used to fit a PCA-based hand model and that acts

as an initial alignment between the template and the raw scan. The final registrations are acquired by applying non-rigid iterative closest point between the aligned template and the raw scan. Additionally, the texture UV maps were transferred from the raw scans to a common UV template by finding the correspondences between the barycentric coordinates of the two meshes. Then, a high fidelity mesh convolutional autoencoder that is based on spiral mesh convolutions was trained and contrasted with state-of-the-art hand models that rely on PCA decomposition. Experimental results show that the graph based autoencoder is able to preserve low frequency details with way less parameters. In a series of experiments, the proposed hand model achieved remarkable reconstruction performance and it is able to fit 3D human hands from diverse age and ethnicity groups.

Next, in Chapter 4 the idea of creating a deformable 3D model using mesh convolutions was extended to dynamic meshes. Until recently, the common practice to model 4D mesh sequences was by interpolating the parameters of a statistical model such as PCA. However, interpolation on the latent space hinders the generated meshes to realistic poses. The proposed method is composed by a LSTM encoder network that autoregressively encodes the expression labels to a latent vector that drives the decoder to the desired expression. Spiral mesh convolutions are utilized for the encoder which, can effectively animate the mesh and enable the creation of realistic and extreme expressions. On the contrary, PCA based decoder fails to create sharp details and extreme deformations when animating the mesh resulting in poor realism of the generated expressions. Importantly, the proposed model, that is based on graph convolutions, experimentally outperformed PCA blendshape model by a large margin in challenging areas such as mouth, cheeks and eyebrows that deform the most during facial expressions. The proposed expression animation model acts directly on the deformation space which ensures the preservation of the subject's identity details. Given that the expression labels can take values over the continuum of each frame, the user can fully-customize the desired generated expression along with its intensity. Finally, the proposed method can be used to animate human expressions directly from images generating realistic expressions that are very similar to the original subjects' expressions.

In Chapter 5, another long studied problem in 3D processing was revisited, namely simplification. A major drawback of 3D models is that they usually require an enormous amount of points to properly describe a surface in high resolution and high fidelity. From modern 3D scanners to sample continuous surfaces, the acquired point clouds contain redundant amount of information making processing, editing and storing point clouds a challenging task. Particularly, this Chapter attempts to tackle the limitations of traditional methods that rely on time consuming iterative optimizations by utilizing graph neural networks. To construct the underlying graph from the point cloud structure, an

efficient k-nn strategy was used. The core of the method proposed, lies on the non-linear projection of the points to high dimensional space that can effectively discriminate salient versus non salient points. Using traditional Farthest Point Sampling on the embedded space, a set of anchor points were sampled in real time while retaining the gradient flow through the projection network. The final component of the proposed network is a refinement module, a graph neural network that modifies the anchor points positions given their respective neighbours to match the appearance of the original high resolution point cloud. To maximize the appearance similarity, a curvature oriented loss function was utilized that has been shown to highly correlate with human perception. The proposed method, apart from real time simplification, can simplify point clouds at any scale simply by modifying the number of anchor points sampled by Farthest Point Sampling. Additionally, the proposed method can generalize well on unseen point clouds avoiding solving time consuming per-cloud optimizations. Compared to traditional methods such as Farthest Point Sampling on xyz-space and Quadric Simplification that produce smooth results with almost uniform sampling, the proposed method preserves much more details of the input even at extreme simplification ratios (i.e. 5% of the original size). Besides its merits and advantages, a limitation of this method however is that Farthest Point Sampling can be very time demanding for high simplification ratios and large-scale point clouds. Additionally, the proposed method can only simplify meshes treated as point clouds and then re-triangulated using off-the-self meshing algorithms, a fact that limits its applicability on meshes.

In Chapter 6, the Point Cloud Simplification method presented in the Chapter 5 was extended, by proposing a neural based mesh simplification method that tackles the limitation of the literature. The devised method relies on an advanced point-sampling module, using a differentiable constant complexity real-time sampling module, based on assigning inclusion probabilities to the vertices of the input mesh based. A relative position GNN is utilized that embeds each vertex on a high dimensional space that captures local surface details, and then assigns an inclusion probability to each one of them. To enable real-time sampling, a multinomial point sampler is settled that samples points according to their inclusion probability. The second part of the proposed network is composed by an edge predictor that constructs an extended, to the original, mesh graph by connecting the sampled vertices with the original vertices sets. Using the candidate edge set, a candidate face set was constructed simply by connecting the triplet of points that share edges. The final module of the proposed method is composed by a Face Classifier that learns to assign an inclusion probability to each formed triangle based on the regularity of the triangle using a novel triangle-graph convolution. The devised triangle convolution, named TriConv, acts directly on a graph constructed between the barycentric coordinates

of each triangle and takes as input the relative features of each triangle face and assigns an inclusion probability to each candidate face. To train the proposed method a combination of loss functions is used, in order to constrain the generated mesh, not only match the appearance of the original mesh, but also to produce a smooth surface on the simplified point cloud. The proposed method is extensively evaluated in a series of extensive experiments that revealed its power to outperform traditional methods under several metrics. Finally, apart from real-time execution, the proposed method has contributed to meshing producing smooth meshes.

Ethical Considerations

Developing a models that are able to generate and animate 3D human hands and 3D facial expressions raises important ethical considerations. Although all of the data used in this thesis have been granted with consent forms from the subjects participated in the data collections, it is necessary to protect their individual rights and prevent potential misuse of the data. Furthermore, given the large diversity of the data used, responsible utilization of these models is essential. Ongoing monitoring and assessment of the network's outputs should be in place to identify and rectify any unintended consequences or ethical implications that may arise from its deployment. It's imperative to use these technologies in a manner that upholds ethical standards and societal well-being, ensuring responsible and accountable practices.

7.1 Future Work

While all of the aforementioned methods can be considered a step forward for 3D shape analysis, modeling and simplification of 3D meshes and point clouds, the work presented in this thesis opens up opportunities for further extensions and improvements in a number of ways:

- Although the hand model presented on Chapter 3 presents a step towards a highly detailed hand modeling it lacks a blend skinning model that deforms the human hand given a target pose. A common issue of linear blend skinning model is the volume loss under certain poses [119, 2]. Traditionally, this issue is mitigated by hand crafted pose that deforms the posed hand to preserve the volume of the original mesh. However, this process requires an artist to carefully design the pose corrective for a wide range of poses and can not be transferred to different templates. A

crucial and important extension of the proposed model would be the creation of a neural skinning module that will pose a hand without the need of hand correctives. This would require a large scale dataset with many subjects that pose different hand expression such as the one acquired in Science Museum London in 2019 that contains over 1400 subjects with a variety of hand poses. The first step to train a hand-model is to encode hand pose information which, demands registered data on a common template. With this modeling approach will manage to overcome posing limitations of current hand models [25, 2] and will completely decompose pose from blend skinning methods. Current approaches implicitly constrain hand poses to be close to the open pose or by hand crafted rotation matrices, making their solutions constrained and sub-optimal, allowing the network to generate irregular poses. By training a network with such a large scale dataset of hand poses, it is feasible to directly construct a strong pose prior that will constrain the pose space to plausible hand poses.

- Recently, diffusion models have emerged as powerful generative models in the field of computer vision achieving outstanding results and providing a powerful tool for image generation [221, 222]. Given their ability to generate sharp and detailed images while overcoming mode collapse issues of GANs, diffusion models have been established as generative models in the field. Although several works have attempted to extend diffusion models to 3D generative task, the field is relatively unexplored. Currently, two methods have been proposed to generate 3D point clouds from noise using the denoising diffusion formulation [223, 224] that attains promising results. However, even though they produce noisy clouds by injecting noise at point level they completely neglect the underlying structure of the point cloud resulting in non-smooth results. Given that graph neural networks tend to generate Laplacian smooth surfaces by their formulation, one could extend traditional 3D modeling by bridging the two worlds of geometric deep learning and diffusion models. In particular, instead of neglecting the connectivity between points, noise could be diffused while at the same time retaining the input connectivity. Subsequently, the denoising process would be formulated with mesh convolutions that respect the topology of the surface. Such method could explore the potential application of denoising diffusion models on both meshes and point clouds provide a powerful alternative to Variational Autoencoders and PCA models. Apart from static 3D deformable models, the potential impact of diffusion models on dynamic mesh generation could be also examined, extending the research presented in Chapter 4.

- Another important direction to explore would be the combination of different modalities apart from 3D meshes, such as speech signals. In particular, a very impactful

application that could arise from the combination of static and dynamic 3D models would be a generative sign language hand model. A lot of effort has been set in the design of models that detect, decode and understand sign language however the generation of hand poses that translate to sign language is particularly unexplored. During the data collection at Science Museum during 2019, a set of sign language poses were performed by volunteers that would set a strong prior for the generation of realistic human sign language. It is extremely challenging, however, to combine generative 3D model with a different modality such as speech or text. Over the last years, several 2D datasets that contain annotated videos of sign language have been developed [225, 226] that would be beneficial in order to translate speech and text to 3D hand poses.

- In Chapter 6, a novel method to simplify meshes and point clouds was presented. By learning to directly triangulate a set of sparse points, without the time-demanding iterative collapsing method, a real-time solution was proposed, that is able to simplify meshes on-the-fly. However, the proposed method cannot guarantee that the resulting meshes will always respect the manifold criteria. Particularly, instead of applying a loss to penalize irregular triangles, a small manifold patches could fitted which will guarantee the that the triangulated points will be manifold. Such method will not only advent the research of neural mesh simplification but also the neural triangulation that has received a lot of attention over the last years.

- In addition to its limitations on the meshing process, the proposed neural simplification method lacks the ability to directly simplify a mesh according to its texture appearance. As shown in Texture-based Quadric Mesh Simplification [59], shape and texture driven simplification result in completely different renderings. Considering that the proposed method is fully differentiable, one can simply fine tune the proposed method with an additional loss function with the aim to preserve the texture of the original mesh. This could be implemented by taking advantage of a differentiable renderer and by maximizing the similarity between the original mesh rendering and the simplified one. The aforementioned extension will benefit many applications such as video games and analysis-by-synthesis that require real-time renderings. By simplifying a highly detailed 3D mesh in real-time, rendering will be extremely accelerated while at the same time the resulting rendering quality will be preserved.

- Apart from the aforementioned limitions presented in this thesis, there is a common major limitation in the field of 3D modeling. In general, 3D models have been traditionally trained on a fixed discretized surface with a pre-specified number of

points and faces. Spiral mesh convolutions exploit the inherited correspondences of the common topology and follow a common pattern across each neighborhood of the mesh. However, the common template requirement prevents from modeling arbitrary topology meshes and usually a lot of effort is required to align and register the meshes in a common template. Recently, some methods have attempted to create 3D morphable models without registered data by using implicit functions [227, 228]. However, implicit functions rely on signed distance fields which cannot be directly edited and visualized in common graphics software tools and usually a meshing step with Marching Cubes [229] is required. A solution to alleviate such issue would be a mesh convolution operator that does not rely on the fixed topology of the mesh but implicitly aligns meshes based on their structural features. Such operator should implicitly order the neighborhood and exploit the intrinsic correspondences of each mesh. With this formulation, mesh convolutions would enable imitation of the behaviour of CNNs without having to explicitly define the ordering of each neighborhood.

BIBLIOGRAPHY

[1] C. Zimmermann, D. Ceylan, J. Yang, B. Russell, M. Argus, and T. Brox, "Freihand: A dataset for markerless capture of hand pose and shape from single rgb images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 813–822.

[2] J. Romero, D. Tzionas, and M. J. Black, "Embodied hands: Modeling and capturing hands and bodies together," *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, vol. 36, no. 6, Nov. 2017.

[3] M. Gao, N. Ruan, J. Shi, and W. Zhou, "Deep neural network for 3d shape classification based on mesh feature," *Sensors*, vol. 22, no. 18, p. 7040, 2022.

[4] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, pp. 18–42, 2017.

[5] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on riemannian manifolds," in *Proceedings of the IEEE international conference on computer vision workshops*, 2015, pp. 37–45.

[6] I. Lim, A. Dielen, M. Campen, and L. Kobbelt, "A simple approach to intrinsic correspondence learning on unstructured 3d meshes," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 349–362.

[7] G. Bouritsas, S. Bokhnyak, S. Ploumpis, M. Bronstein, and S. Zafeiriou, "Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learn-

ing and generation," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[8] N. Qian, J. Wang, F. Mueller, F. Bernard, V. Golyanik, and C. Theobalt, "HTML: A Parametric Hand Texture Model for 3D Hand Reconstruction and Personalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020.

[9] B. Gecer, S. Ploumpis, I. Kotsia, and S. Zafeiriou, "Ganfit: Generative adversarial network fitting for high fidelity 3d face reconstruction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1155–1164.

[10] R. A. Potamias, G. Bouritsas, and S. Zafeiriou, "Revisiting point cloud simplification: A learnable feature preserving approach," in *European Conference on Computer Vision*. Springer, 2022, pp. 586–603.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[13] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[14] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[15] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[16] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," *arXiv preprint arXiv:2304.02643*, 2023.

[17] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2015, pp. 922–928.

[18] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[19] Z. Liu, H. Tang, Y. Lin, and S. Han, "Point-voxel cnn for efficient 3d deep learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[20] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[21] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," *Advances in neural information processing systems*, vol. 30, 2017.

[22] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[23] R. A. Potamias, J. Zheng, S. Ploumpis, G. Bouritsas, E. Ververas, and S. Zafeiriou, "Learning to generate customized dynamic 3d facial expressions," in *European Conference on Computer Vision*. Springer, 2020, pp. 278–294.

[24] R. A. Potamias, S. Ploumpis, and S. Zafeiriou, "Neural mesh simplification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 583–18 592.

[25] R. A. Potamias, S. Ploumpis, S. Moschoglou, V. Triantafyllou, and S. Zafeiriou, "Handy: Towards a high fidelity 3d hand shape and appearance model," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023.

[26] R. A. Potamias, A. Neofytou, K. M. Bintsi, and S. Zafeiriou, "Graphwalks: efficient shape agnostic geodesic shortest path estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 2968–2977.

[27] K.-M. Bintsi, V. Baltatzis, R. A. Potamias, A. Hammers, and D. Rueckert, "Multimodal brain age estimation using interpretable adaptive population-graph learning," *arXiv preprint arXiv:2307.04639*, 2023.

[28] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, pp. 83–98, 2013.

[29] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[30] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[31] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[32] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[33] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks," in *Advances in neural information processing systems*, 2016, pp. 3189–3197.

[34] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[35] M. Fey, J. E. Lenssen, F. Weichert, and H. Müller, "Splinecnn: Fast geometric deep learning with continuous b-spline kernels," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 869–877.

[36] N. Verma, E. Boyer, and J. Verbeek, "Feastnet: Feature-steered graph convolutions for 3d shape analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2598–2606.

[37] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017.

[38] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=ryGs6iA5Km

[39] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics (TOG)*, 2019.

[40] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," *CoRR*, 2017.

[41] Z. Zhang, B.-S. Hua, and S.-K. Yeung, "Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics," in *International Conference on Computer Vision (ICCV)*, 2019.

[42] H. Lei, N. Akhtar, and A. Mian, "Octree guided cnn with spherical kernels for 3d point clouds," *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[43] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "Spidercnn: Deep learning on point sets with parameterized convolutional filters," *arXiv preprint arXiv:1803.11527*, 2018.

[44] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 11, pp. 1944–1957, 2007.

[45] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016.

[46] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.

[47] C. Cangea, P. Veličković, N. Jovanović, T. Kipf, and P. Liò, "Towards sparse hierarchical graph classifiers," *arXiv preprint arXiv:1811.01287*, 2018.

[48] H. Gao and S. Ji, "Graph u-nets," in *International Conference on Machine Learning*, 2019, pp. 2083–2092.

[49] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 3734–3743.

[50] E. Ranjan, S. Sanyal, and P. P. Talukdar, "ASAP: Adaptive structure aware pooling for learning hierarchical graph representations," *arXiv preprint arXiv:1911.07979*, 2019.

[51] M. I. K. Islam, M. Khanov, and E. Akbas, "Mpool: Motif-based graph pooling," *arXiv preprint arXiv:2303.03654*, 2023.

[52] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen, "Decimation of triangle meshes," in *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, 1992, pp. 65–70.

[53] J. Rossignac and P. Borrel, "Multi-resolution 3d approximations for rendering complex scenes," in *Modeling in computer graphics*. Springer, 1993, pp. 455–465.

[54] W. J. Schroeder, "A topology modifying progressive decimation algorithm," in *Proceedings. Visualization'97 (Cat. No. 97CB36155)*. IEEE, 1997, pp. 205–212.

[55] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," in *Proceedings of the 19th annual conference on computer graphics and interactive techniques*, 1992, pp. 71–78.

[56] H. Hoppe, "Progressive meshes," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 99–108.

[57] R. Ronfard and J. Rossignac, "Full-range approximation of triangulated polyhedra." in *Computer Graphics Forum*, vol. 15. Wiley Online Library, 1996, pp. 67–76.

[58] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 1997, pp. 209–216.

[59] ——, "Simplifying surfaces with color and texture using quadric error metrics," in *Proceedings Visualization'98 (Cat. No. 98CB36276)*. IEEE, 1998, pp. 263–269.

[60] K. Bahirat, C. Lai, R. P. Mcmahan, and B. Prabhakaran, "Designing and evaluating a mesh simplification algorithm for virtual reality," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 14, no. 3s, Jun. 2018.

[61] S. Kim, W. Jeong, and C. Kim, "Lod generation with discrete curvature error metric," in *Proceedings of Korea Israel Bi-National Conference*. Citeseer, 1999, pp. 97–104.

[62] S.-J. Kim, C.-H. Kim, and D. Levin, "Surface simplification using a discrete curvature norm," *Computers & Graphics*, vol. 26, no. 5, pp. 657–663, 2002.

[63] X.-L. Liu, Z.-Y. Liu, P.-D. Gao, and X. Peng, "Edge collapse simplification based on sharp degree." *Ruan Jian Xue Bao(J. Softw.)*, vol. 16, no. 5, pp. 669–675, 2005.

[64] L. Yao, S. Huang, H. Xu, and P. Li, "Quadratic error metric mesh simplification algorithm based on discrete curvature," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[65] J. Wu, X. Shen, W. Zhu, and L. Liu, "Mesh saliency with global rarity," *Graphical Models*, vol. 75, no. 5, pp. 255–264, 2013.

[66] G. An, T. Watanabe, and M. Kakimoto, "Mesh simplification using hybrid saliency," in *2016 International Conference on Cyberworlds (CW)*. IEEE, 2016, pp. 231–234.

[67] W. Luan, C. Liu, and H. Pang, "Skeleton-bridged mesh saliency and mesh simplification," in *2020 4th International Conference on Computer Science and Artificial Intelligence*, 2020, pp. 278–283.

[68] H.-T. D. Liu, A. Jacobson, and M. Ovsjanikov, "Spectral coarsening of geometric operators," *ACM Trans. Graph.*, vol. 38, no. 4, Jul. 2019.

[69] T. Lescoat, H.-T. D. Liu, J.-M. Thiery, A. Jacobson, T. Boubekeur, and M. Ovsjanikov, "Spectral mesh simplification," *Computer Graphics Forum*, vol. 39, no. 2, pp. 315–324, 2020.

[70] A. Papageorgiou and N. Platis, "Triangular mesh simplification on the gpu," *The Visual Computer*, vol. 31, no. 2, pp. 235–244, 2015.

[71] H. Lee and M.-H. Kyung, "Parallel mesh simplification using embedded tree collapsing," *The Visual Computer*, vol. 32, no. 6, pp. 967–976, 2016.

[72] D. Cohen-Steiner, P. Alliez, and M. Desbrun, "Variational shape approximation," in *ACM SIGGRAPH 2004 Papers*, 2004, pp. 905–914.

[73] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "Meshcnn: A network with an edge," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 90:1–90:12, 2019.

[74] R. Sanchez-Reillo, C. Sanchez-Avila, and A. Gonzalez-Marcos, "Biometric identification through hand geometry measurements," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 10, pp. 1168–1171, 2000.

[75] N. Duta, "A survey of biometric technology based on hand shape," *Pattern recognition*, vol. 42, no. 11, pp. 2797–2806, 2009.

[76] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun, "Realtime and robust hand tracking from depth," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1106–1113.

[77] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei *et al.*, "Accurate, robust, and flexible real-time hand tracking," in *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, 2015, pp. 3633–3642.

[78] L. Jiang, H. Xia, and C. Guo, "A model-based system for real-time articulated hand tracking using a simple data glove and a depth camera," *Sensors*, vol. 19, no. 21, p. 4680, 2019.

[79] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "Mediapipe hands: On-device real-time hand tracking," *arXiv preprint arXiv:2006.10214*, 2020.

[80] U. Iqbal, P. Molchanov, T. B. J. Gall, and J. Kautz, "Hand pose estimation via latent 2.5 d heatmap regression," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 118–134.

[81] E. Ng, S. Ginosar, T. Darrell, and H. Joo, "Body2hands: Learning to infer 3d hands from conversational gesture body dynamics," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 865–11 874.

[82] M. de La Gorce, N. Paragios, and D. J. Fleet, "Model-based hand tracking with texture, shading and self-occlusions," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–8.

[83] M. de La Gorce, D. J. Fleet, and N. Paragios, "Model-based 3d hand pose estimation from monocular video," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 9, pp. 1793–1805, 2011.

[84] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid, "Learning from synthetic humans," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 109–117.

[85] E. Corona, T. Hodan, M. Vo, F. Moreno-Noguer, C. Sweeney, R. Newcombe, and L. Ma, "Lisa: Learning implicit shape and appearance of hands," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 20 533–20 543.

[86] L. Yang, K. Li, X. Zhan, J. Lv, W. Xu, J. Li, and C. Lu, "Artiboost: Boosting articulated 3d hand-object pose estimation via online exploration and synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 2750–2760.

[87] A. Boukhayma, R. d. Bem, and P. H. Torr, "3d hand shape and pose from images in the wild," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 843–10 852.

[88] Y. Hasson, G. Varol, D. Tzionas, I. Kalevatykh, M. J. Black, I. Laptev, and C. Schmid, "Learning joint reconstruction of hands and manipulated objects," in *CVPR*, 2019.

[89] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect." in *BmVC*, vol. 1, no. 2, 2011, p. 3.

[90] S. Sridhar, H. Rhodin, H.-P. Seidel, A. Oulasvirta, and C. Theobalt, "Real-time hand tracking using a sum of anisotropic gaussians model," in *2014 2nd International Conference on 3D Vision*, vol. 1. IEEE, 2014, pp. 319–326.

[91] A. Tkach, M. Pauly, and A. Tagliasacchi, "Sphere-meshes for real-time hand modeling and tracking," *ACM Transactions on Graphics (ToG)*, vol. 35, no. 6, pp. 1–11, 2016.

[92] S. Melax, L. Keselman, and S. Orsten, "Dynamics based 3d skeletal hand tracking," in *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2013, pp. 184–184.

[93] T. Schmidt, R. A. Newcombe, and D. Fox, "Dart: Dense articulated real-time tracking." in *Robotics: Science and Systems*, vol. 2, no. 1. Berkeley, CA, 2014, pp. 1–9.

[94] S. Khamis, J. Taylor, J. Shotton, C. Keskin, S. Izadi, and A. Fitzgibbon, "Learning an efficient model of hand shape variation from depth images," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 2540–2548.

[95] Y. Li, L. Zhang, Z. Qiu, Y. Jiang, N. Li, Y. Ma, Y. Zhang, L. Xu, and J. Yu, "Nimble: a non-rigid hand model with bones and muscles," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–16, 2022.

[96] T. Heap and D. Hogg, "Towards 3d hand tracking using a deformable model," in *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*. Ieee, 1996, pp. 140–145.

[97] H. Hamer, K. Schindler, E. Koller-Meier, and L. Van Gool, "Tracking a hand manipulating an object," in *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 1475–1482.

[98] J. Taylor, R. Stebbing, V. Ramakrishna, C. Keskin, J. Shotton, S. Izadi, A. Hertzmann, and A. Fitzgibbon, "User-specific hand modeling from monocular depth sequences," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 644–651.

[99] M. Oberweger, P. Wohlhart, and V. Lepetit, "Training a feedback loop for hand pose estimation," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3316–3324.

[100] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 1145–1153.

[101] G. Pavlakos, L. Zhu, X. Zhou, and K. Daniilidis, "Learning to estimate 3d human pose and shape from a single color image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 459–468.

[102] P. Panteleris, I. Oikonomidis, and A. Argyros, "Using a single rgb frame for real time 3d hand pose estimation in the wild," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 436–445.

[103] C. Zimmermann and T. Brox, "Learning to estimate 3d hand pose from single rgb images," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 4903–4911.

[104] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt, "Ganerated hands for real-time 3d hand tracking from monocular rgb," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 49–59.

[105] S. Baek, K. I. Kim, and T.-K. Kim, "Pushing the envelope for rgb-based dense 3d hand pose estimation via neural rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1067–1076.

[106] X. Zhang, Q. Li, H. Mo, W. Zhang, and W. Zheng, "End-to-end hand mesh recovery from a monocular rgb image," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2354–2364.

[107] L. Ge, Z. Ren, Y. Li, Z. Xue, Y. Wang, J. Cai, and J. Yuan, "3d hand shape and pose estimation from a single rgb image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 833–10 842.

[108] D. Kulon, H. Wang, R. A. Güler, M. M. Bronstein, and S. Zafeiriou, "Single image 3d hand reconstruction with mesh convolutions," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2019.

[109] D. Kulon, R. A. Guler, I. Kokkinos, M. M. Bronstein, and S. Zafeiriou, "Weakly-supervised mesh-convolutional hand reconstruction in the wild," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4990–5000.

[110] T. Groueix, M. Fisher, V. G. Kim, B. Russell, and M. Aubry, "AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[111] K. Karunratanakul, J. Yang, Y. Zhang, M. J. Black, K. Muandet, and S. Tang, "Grasping field: Learning implicit representations for human grasps," in *2020 International Conference on 3D Vision (3DV)*. IEEE, 2020, pp. 333–344.

[112] Y. Ye, A. Gupta, and S. Tulsiani, "What's in your hands? 3d reconstruction of generic objects in hands," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3895–3905.

[113] A. Armagan, G. Garcia-Hernando, S. Baek, S. Hampali, M. Rad, Z. Zhang, S. Xie, M. Chen, B. Zhang, F. Xiong *et al.*, "Measuring generalisation to unseen viewpoints, articulations, shapes and objects for 3d hand pose estimation under hand-object interaction," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*. Springer, 2020, pp. 85–101.

[114] Z. Cao, I. Radosavovic, A. Kanazawa, and J. Malik, "Reconstructing hand-object interactions in the wild," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 417–12 426.

[115] Z. Chen, Y. Hasson, C. Schmid, and I. Laptev, "Alignsdf: Pose-aligned signed distance fields for hand-object reconstruction," in *European Conference on Computer Vision*. Springer, 2022, pp. 231–248.

[116] E. Wood, T. Baltrušaitis, C. Hewitt, S. Dziadzio, T. J. Cashman, and J. Shotton, "Fake it till you make it: face analysis in the wild using synthetic data alone," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 3681–3691.

[117] M. Fabbri, G. Brasó, G. Maugeri, O. Cetintas, R. Gasparini, A. Ošep, S. Calderara, L. Leal-Taixé, and R. Cucchiara, "Motsynth: How can synthetic data help pedestrian detection and tracking?" in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 849–10 859.

[118] X. Chen, Y. Liu, Y. Dong, X. Zhang, C. Ma, Y. Xiong, Y. Zhang, and X. Guo, "Mobrecon: Mobile-friendly hand mesh reconstruction from monocular image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 20 544–20 554.

[119] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "Smpl: A skinned multi-person linear model," *ACM transactions on graphics (TOG)*, vol. 34, no. 6, pp. 1–16, 2015.

[120] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[121] B. Amberg, S. Romdhani, and T. Vetter, "Optimal step nonrigid icp algorithms for surface registration," in *2007 IEEE conference on computer vision and pattern recognition*. IEEE, 2007, pp. 1–8.

[122] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black, "Generating 3d faces using convolutional mesh autoencoders," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 704–720.

[123] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila, "Alias-free generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 852–863, 2021.

[124] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[125] T. Bolkart and S. Wuhrer, "A groupwise multilinear correspondence optimization for 3d faces," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3604–3612.

[126] L. Guan Ming, J. Prayook, and A. Wei Tech, "Mobilehand: Real-time 3d hand shape and pose estimation from color image," in *27th International Conference on Neural Information Processing (ICONIP)*, 2020.

[127] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018.

[128] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4401–4410.

[129] Y. Chen, Z. Tu, D. Kang, L. Bao, Y. Zhang, X. Zhe, R. Chen, and J. Yuan, "Model-based 3d hand reconstruction via self-supervised learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 451–10 460.

[130] J. Ren, J. Zhu, and J. Zhang, "End-to-end weakly-supervised multiple 3d hand mesh reconstruction from single image," *arXiv preprint arXiv:2204.08154*, 2022.

[131] S. Cheng, I. Kotsia, M. Pantic, and S. Zafeiriou, "4dfab: A large scale 4d database for facial expression analysis and biometric applications," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5117–5126.

[132] S. Ploumpis, H. Wang, N. Pears, W. A. Smith, and S. Zafeiriou, "Combining 3d morphable models: A large scale face-and-head model," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 934–10 943.

[133] C. Cao, Q. Hou, and K. Zhou, "Displaced dynamic expression regression for real-time facial tracking and animation," *ACM Transactions on graphics (TOG)*, pp. 1–10, 2014.

[134] S. Ploumpis, E. Ververas, E. O. Sullivan, S. Moschoglou, H. Wang, N. Pears, W. A. Smith, B. Gecer, and S. Zafeiriou, "Towards a complete 3d morphable model of the human head," *ArXiv*, 2019.

[135] C.-K. Yang and W.-T. Chiang, "An interactive facial expression generation system," *Multimedia Tools and Applications*, pp. 41–60, 2008.

[136] Y. Zhou and B. E. Shi, "Photorealistic facial expression synthesis by the conditional difference adversarial autoencoder," in *2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII)*. IEEE, 2017, pp. 370–376.

[137] L. Fan, W. Huang, C. Gan, J. Huang, and B. Gong, "Controllable image-to-video translation: A case study on facial expression generation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, pp. 3510–3517.

[138] N. Otberdout, M. Daoudi, A. Kacem, L. Ballihi, and S. Berretti, "Dynamic facial expression generation on hilbert hypersphere with conditional wasserstein generative adversarial nets," *ArXiv*, 2019.

[139] T. Karras, T. Aila, S. Laine, A. Herva, and J. Lehtinen, "Audio-driven facial animation by joint end-to-end learning of pose and emotion," *ACM Transactions on Graphics (TOG)*, p. 94, 2017.

[140] D. Cudeiro, T. Bolkart, C. Laidlaw, A. Ranjan, and M. J. Black, "Capture, learning, and synthesis of 3D speaking styles," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 101–10 111.

[141] J. Thies, M. Zollhöfer, M. Nießner, L. Valgaerts, M. Stamminger, and C. Theobalt, "Real-time expression transfer for facial reenactment." *ACM Trans. Graph.*, vol. 34, no. 6, pp. 183–1, 2015.

[142] B. Amberg, R. Knothe, and T. Vetter, "Expression invariant 3d face recognition with a morphable model," in *2008 8th IEEE International Conference on Automatic Face Gesture Recognition*, 2008, pp. 1–6.

[143] F. Yang, J. Wang, E. Shechtman, L. Bourdev, and D. Metaxas, "Expression flow for 3d-aware face component transfer," in *ACM SIGGRAPH 2011 papers*, 2011, pp. 1–10.

[144] S. Bouaziz, Y. Wang, and M. Pauly, "Online modeling for realtime facial animation," *ACM Transactions on Graphics (ToG)*, vol. 32, no. 4, pp. 1–10, 2013.

[145] Y. Li, M. R. Min, D. Shen, D. Carlson, and L. Carin, "Video Generation From Text," *ArXiv*, 2017.

[146] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *ArXiv*, 2014.

[147] H. X. Pham, S. Cheung, and V. Pavlovic, "Speech-Driven 3D Facial Animation with Implicit Emotional Awareness: A Deep Learning Approach," in *30th IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPRW 2017*, ser. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops. United States: IEEE Computer Society, 2017, pp. 2328–2336.

[148] H. X. Pham, Y. Wang, and V. Pavlovic, "End-to-end learning for 3d facial animation from speech," in *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, ser. ICMI '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 361–365. [Online]. Available: https://doi.org/10.1145/3242969.3243017

[149] P. Tzirakis, A. Papaioannou, A. Lattas, M. Tarasiou, B. W. Schuller, and S. Zafeiriou, "Synthesising 3D Facial Motion from "In-the-Wild" Speech," *CoRR*, 2019.

[150] G. Tian, Y. Yuan, and Y. Liu, "Audio2face: Generating speech/face animation from single audio with attention-based bidirectional lstm networks," in *2019 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2019, pp. 366–371.

[151] N. Otberdout, C. Ferrari, M. Daoudi, S. Berretti, and A. Del Bimbo, "Sparse to dense dynamic 3d facial expression generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 20 385–20 394.

[152] C. Zhong, Z. Sun, and T. Tan, "Robust 3d face recognition using learned visual codebook," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–6.

[153] A. Savran, N. Alyüz, H. Dibeklioğlu, O. Çeliktutan, B. Gökberk, B. Sankur, and L. Akarun, "Bosphorus database for 3d face analysis," in *European Workshop on Biometrics and Identity Management*. Springer, 2008, pp. 47–56.

[154] G. Stratou, A. Ghosh, P. Debevec, and L.-P. Morency, "Effect of illumination on automatic expression recognition: a novel 3d relightable facial database," in *Face and Gesture 2011*. IEEE, 2011, pp. 611–618.

[155] A. Moreno, "Gavabdb: a 3d face database," in *Proc. 2nd COST275 Workshop on Biometrics on the Internet, 2004*, 2004, pp. 75–80.

[156] S. Gupta, M. K. Markey, and A. C. Bovik, "Anthropometric 3d face recognition," *International journal of computer vision*, pp. 331–349, 2010.

[157] H. Yang, H. Zhu, Y. Wang, M. Huang, Q. Shen, R. Yang, and X. Cao, "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*, 2020, pp. 601–610.

[158] G. Papaioannou, Athanasios an Baris, S. Cheng, G. G. Chrysos, J. Deng, E. Fotiadou, C. Kampouris, D. Kollias, S. Moschoglou, K. Songsri-In, S. Ploumpis, G. Trigeorgis, P. Tzirakis, E. Ververas, Y. Zhou, A. Ponniah, A. Roussos, and S. Zafeiriou, "Mimicme: A large scale diverse 4d database for facial expression analysis," in *Proceedings of the European Conference on Computer Vision*, 2022.

[159] D. Cosker, E. Krumhuber, and A. Hilton, "A facs valid 3d dynamic action unit database with applications to 3d dynamic morphable facial modeling," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 2296–2303.

[160] Y. Chang, M. Vieira, M. Turk, and L. Velho, "Automatic 3d facial expression analysis in videos," in *International Workshop on Analysis and Modeling of Faces and Gestures*. Springer, 2005, pp. 293–307.

[161] L. Yin, X. C. Y. Sun, T. Worm, and M. Reale, "A high-resolution 3d dynamic facial expression database, 2008," in *IEEE International Conference on Automatic Face and Gesture Recognition, Amsterdam, The Netherlands*.

[162] T. Alashkar, B. Ben Amor, M. Daoudi, and S. Berretti, "A 3D Dynamic Database for Unconstrained Face Recognition," in *5th International Conference and Exhibition on 3D Body Scanning Technologies*, Lugano, Switzerland, Oct. 2014.

[163] G. Fanelli, J. Gall, H. Romsdorfer, T. Weise, and L. van Gool, "A 3d audio-visual corpus of affective communication," *IEEE MultiMedia*, no. 6, 2010.

[164] X. Zhang, L. Yin, J. F. Cohn, S. Canavan, M. Reale, A. Horowitz, and P. Liu, "A high-resolution spontaneous 3d dynamic facial expression database," in *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*. IEEE, 2013, pp. 1–6.

[165] Z. Zhang, J. M. Girard, Y. Wu, X. Zhang, P. Liu, U. Ciftci, S. Canavan, M. Reale, A. Horowitz, H. Yang *et al.*, "Multimodal spontaneous emotion corpus for human behavior analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3438–3446.

[166] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010, pp. 807–814.

[167] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[168] T. Lu and T.-H. Chao, "A single-camera system captures high-resolution 3d images in one shot," *SPIE Newsroom*, 2006.

[169] A. Lattas, S. Moschoglou, B. Gecer, S. Ploumpis, V. Triantafyllou, A. Ghosh, and S. Zafeiriou, "Avatarme: Realistically renderable 3d facial reconstruction" in-the-wild"," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 760–769.

[170] G. Pavlidis, A. Koutsoudis, F. Arnaoutoglou, V. Tsioukas, and C. Chamzas, "Methods for 3d digitization of cultural heritage," *Journal of cultural heritage*, vol. 8, no. 1, pp. 93–98, 2007.

[171] P. Cignoni, C. Montani, and R. Scopigno, "A comparison of mesh simplification algorithms," *Computers & Graphics*, vol. 22, no. 1, pp. 37–54, 1998.

[172] C. H. Lee, A. Varshney, and D. W. Jacobs, "Mesh saliency," in *ACM SIGGRAPH 2005 Papers*, 2005, pp. 659–666.

[173] G. Lavoué, "A local roughness measure for 3d meshes and its application to visual masking," *ACM Transactions on Applied perception (TAP)*, vol. 5, no. 4, pp. 1–23, 2009.

[174] M. Pauly, M. Gross, and L. P. Kobbelt, "Efficient simplification of point-sampled surfaces," in *IEEE Visualization, 2002. VIS 2002.* IEEE, 2002, pp. 163–170.

[175] J. Qi, W. Hu, and Z. Guo, "Feature preserving and uniformity-controllable point cloud simplification on graph," in *2019 IEEE International Conference on Multimedia and Expo (ICME).* IEEE, 2019, pp. 284–289.

[176] B.-Q. Shi, J. Liang, and Q. Liu, "Adaptive simplification of point cloud using k-means clustering," *Computer-Aided Design*, vol. 43, no. 8, pp. 910–922, 2011.

[177] H. Han, X. Han, F. Sun, and C. Huang, "Point cloud simplification with preserved edge based on normal vector," *Optik-International Journal for Light and Electron Optics*, vol. 126, no. 19, pp. 2157–2162, 2015.

[178] K. Zhang, S. Qiao, X. Wang, Y. Yang, and Y. Zhang, "Feature-preserved point cloud simplification based on natural quadric shape models," *Applied Sciences*, vol. 9, no. 10, p. 2130, 2019.

[179] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Point set surfaces," in *Proceedings Visualization, 2001. VIS'01.* IEEE, 2001, pp. 21–29.

[180] L. M. Galantucci and G. Percoco, "A multilevel approach to edge detection in tessellated point clouds," *CIRP annals*, vol. 54, no. 1, pp. 127–130, 2005.

[181] N. Leal, E. Leal, and S.-T. German, "A linear programming approach for 3d point cloud simplification," *IAENG International Journal of Computer Science*, vol. 44, no. 1, pp. 60–67, 2017.

[182] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi, "The farthest point strategy for progressive image sampling," *IEEE Transactions on Image Processing*, vol. 6, no. 9, pp. 1305–1315, 1997.

[183] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui, "Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5589–5598.

[184] O. Dovrat, I. Lang, and S. Avidan, "Learning to sample," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2760–2769.

[185] I. Lang, A. Manor, and S. Avidan, "Samplenet: differentiable point cloud sampling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7578–7588.

[186] G. Lavoué and M. Corsini, "A comparison of perceptually-based metrics for objective evaluation of geometry processing," *IEEE Transactions on Multimedia*, vol. 12, no. 7, pp. 636–649, 2010.

[187] M. Corsini, M.-C. Larabi, G. Lavoué, O. Petřík, L. Váša, and K. Wang, "Perceptual metrics for static and dynamic triangle meshes," in *Computer Graphics Forum*, vol. 32, no. 1. Wiley Online Library, 2013, pp. 101–125.

[188] Z. Karni and C. Gotsman, "Spectral compression of mesh geometry," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000, pp. 279–286.

[189] S.-J. Kim, S.-K. Kim, and C.-H. Kim, "Discrete differential error metric for surface simplification," in *10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings.* IEEE, 2002, pp. 276–283.

[190] G. Lavoué, "A multiscale metric for 3d mesh visual quality assessment," in *Computer Graphics Forum*, vol. 30, no. 5. Wiley Online Library, 2011, pp. 1427–1437.

[191] F. Torkhani, K. Wang, and J.-M. Chassery, "A curvature tensor distance for mesh visual quality assessment," in *International Conference on Computer Vision and Graphics.* Springer, 2012, pp. 253–263.

[192] L. Váša and J. Rus, "Dihedral angle mesh error: a fast perception correlated distortion measure for fixed connectivity triangle meshes," in *Computer Graphics Forum*, vol. 31, no. 5. Wiley Online Library, 2012, pp. 1715–1724.

[193] G. Lavoué, I. Cheng, and A. Basu, "Perceptual quality metrics for 3d meshes: towards an optimal multi-attribute computational model," in *2013 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2013, pp. 3271–3276.

[194] L. Dong, Y. Fang, W. Lin, and H. S. Seah, "Perceptual quality assessment for 3d triangle mesh based on curvature," *IEEE Transactions on Multimedia*, vol. 17, no. 12, pp. 2174–2184, 2015.

[195] X. Feng, W. Wan, R. Y. Da Xu, H. Chen, P. Li, and J. A. Sánchez, "A perceptual quality metric for 3d triangle meshes based on spatial pooling," *Frontiers of Computer Science*, vol. 12, no. 4, pp. 798–812, 2018.

[196] Z. C. Yildiz, A. C. Oztireli, and T. Capin, "A machine learning framework for full-reference 3d shape quality assessment," *The Visual Computer*, vol. 36, no. 1, pp. 127–139, 2020.

[197] R. A. Potamias, A. Neofytou, K. M. Bintsi, and S. Zafeiriou, "Graphwalks: Efficient shape agnostic geodesic shortest path estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2022, pp. 2968–2977.

[198] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, 2017.

[199] J. Jin, A. G. Patil, Z. Xiong, and H. Zhang, "Dr-kfs: A differentiable visual similarity metric for 3d shape reconstruction," in *European Conference on Computer Vision*. Springer, 2020, pp. 295–311.

[200] C.-L. Li, T. Simon, J. Saragih, B. Póczos, and Y. Sheikh, "Lbs autoencoder: Self-supervised fitting of articulated meshes to point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 967–11 976.

[201] C. Wen, Y. Zhang, Z. Li, and Y. Fu, "Pixel2mesh++: Multi-view 3d mesh generation via deformation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1042–1051.

[202] Y. Wang, J. Zheng, and H. Wang, "Fast mesh simplification method for three-dimensional geometric models with feature-preserving efficiency," *Scientific Programming*, vol. 2019, 2019.

[203] G. Lavoué, E. D. Gelasca, F. Dupont, A. Baskurt, and T. Ebrahimi, "Perceptually driven 3d distance metrics with application to watermarking," in *Applications of Digital Image Processing XXIX*, vol. 6312. International Society for Optics and Photonics, 2006, p. 63120L.

[204] A. Nasikun, C. Brandt, and K. Hildebrandt, "Fast approximation of laplace-beltrami eigenproblems," in *Computer Graphics Forum*, vol. 37, no. 5. Wiley Online Library, 2018, pp. 121–134.

[205] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, *Numerical geometry of non-rigid shapes*. Springer Science & Business Media, 2008.

[206] J. Booth, A. Roussos, A. Ponniah, D. Dunaway, and S. Zafeiriou, "Large scale 3d morphable models," *International Journal of Computer Vision*, vol. 126, no. 2, pp. 233–254, 2018.

[207] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE transactions on visualization and computer graphics*, vol. 5, no. 4, pp. 349–359, 1999.

[208] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "Meshlab: an open-source mesh processing tool." in *Eurographics Italian chapter conference*, vol. 2008. Salerno, Italy, 2008, pp. 129–136.

[209] W. Tan, N. Qin, L. Ma, Y. Li, J. Du, G. Cai, K. Yang, and J. Li, "Toronto-3d: A large-scale mobile lidar dataset for semantic segmentation of urban roadways," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 202–203.

[210] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 448–456.

[211] A. Dai and M. Nießner, "Scan2mesh: From unstructured range scans to 3d meshes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5574–5583.

[212] N. Sharp and M. Ovsjanikov, "Pointtrinet: Learned triangulation of 3d point sets," in *European Conference on Computer Vision*. Springer, 2020, pp. 762–778.

[213] M.-J. Rakotosaona, P. Guerrero, N. Aigerman, N. J. Mitra, and M. Ovsjanikov, "Learning delaunay surface elements for mesh reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 22–31.

[214] M.-J. Rakotosaona, N. Aigerman, N. Mitra, M. Ovsjanikov, and P. Guerrero, "Differentiable surface triangulation," *arXiv preprint arXiv:2109.10695*, 2021.

[215] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Randla-net: Efficient semantic segmentation of large-scale point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 108–11 117.

[216] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5939–5948.

[217] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 165–174.

[218] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4460–4470.

[219] Y. Liao, S. Donne, and A. Geiger, "Deep marching cubes: Learning explicit surface representations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2916–2925.

[220] M. Liu, X. Zhang, and H. Su, "Meshing point clouds with predicted intrinsic-extrinsic ratio guidance," in *European Conference on Computer Vision*. Springer, 2020, pp. 68–84.

[221] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

[222] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[223] S. Luo and W. Hu, "Diffusion probabilistic models for 3d point cloud generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.

[224] X. Zeng, A. Vahdat, F. Williams, Z. Gojcic, O. Litany, S. Fidler, and K. Kreis, "Lion: Latent point diffusion models for 3d shape generation," *arXiv preprint arXiv:2210.06978*, 2022.

[225] G. Varol, L. Momeni, S. Albanie, T. Afouras, and A. Zisserman, "Read and attend: Temporal localisation in sign language videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 857–16 866.

[226] A. Duarte, S. Palaskar, L. Ventura, D. Ghadiyaram, K. DeHaan, F. Metze, J. Torres, and X. Giro-i Nieto, "How2Sign: A Large-scale Multimodal Dataset for Continuous American Sign Language," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[227] T. Yenamandra, A. Tewari, F. Bernard, H.-P. Seidel, M. Elgharib, D. Cremers, and C. Theobalt, "i3dmm: Deep implicit 3d morphable model of human heads," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 803–12 813.

[228] S. Giebenhain, T. Kirschstein, M. Georgopoulos, M. Rünz, L. Agapito, and M. Nießner, "Learning neural parametric head models," 2022.

[229] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *ACM siggraph computer graphics*, vol. 21, no. 4, pp. 163–169, 1987.