Representation learning of 3D meshes using an Autoencoder in the spectral domain

Clément Lemeunier, Florence Denis, Guillaume Lavoué, Florent Dupont

Computers & Graphics (2022)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Representation learning of 3D meshes using an Autoencoder in the spectral domain

Clément Lemeunier[a], Florence Denis[b], Guillaume Lavoué[c], Florent Dupont[b]

[a]CNRS: Univ Lyon, CNRS, INSA Lyon, UCBL, LIRIS, UMR5205, F-69622 Villeurbanne, France
[b]Lyon 1: Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, UMR5205, F-69622 Villeurbanne, France
[c]Univ Lyon, Centrale Lyon, CNRS, INSA Lyon, UCBL, LIRIS, UMR5205, F-69130 Ecully, France

ARTICLE INFO

Article history:
Received July 7, 2022

Keywords: Geometric Deep Learning, Spectral analysis, Autoencoder, Human body triangular meshes

ABSTRACT

Learning on surfaces is a difficult task: the data being non-Euclidean makes the transfer of known techniques such as convolutions and pooling non trivial. Common methods deploy processes to apply deep learning operations to triangular meshes either in the spatial domain by defining weights between nodes, or in the spectral domain using first order Chebyshev polynomials followed by a return in the spatial domain. In this study, we present a Spectral Autoencoder (SAE) enabling the application of deep learning techniques to 3D meshes by directly giving spectral coefficients obtained with a spectral transform as inputs. With a dataset composed of surfaces having the same connectivity, it is possible with the Graph Laplacian to express the geometry of all samples in the frequency domain. Then, by using an Autoencoder architecture, we are able to extract important features from spectral coefficients without going back to the spatial domain. Finally, a latent space is built from which reconstruction and interpolation is possible. This method allows the treatment of meshes with more vertices by keeping the same architecture, and allows to learn on big datasets with short computation times. Through experiments, we demonstrate that this architecture is able to give better results than state of the art methods in a faster way.

1. Introduction

Recently, acquiring methods like motion capture have become more affordable and have therefore increased publicly available scans. It is now possible to digitize moving shapes such as body or faces while keeping the pose and appearance information at high spatial and temporal resolution. There is today a need to develop models able to treat the information coming from those scans that are most of the time represented as unordered point clouds.

Most common types of deep learning techniques such as convolutions, pooling and up sampling enable to generalize learned weights to unseen data in order to classify, segment or reconstruct from a latent space. Convolutional Autoencoders are useful tools to extract important features from observed samples in an unsupervised way. By forcing the input to go through a bottleneck, the network is able to construct a latent space representing faithfully the manifold of the input samples like all the possible poses of a human body for example. This interesting property offers a way to generate new data by interpolating in the latent space. When learning on images, the use of convolutions is well defined, since the domain has a Euclidean structure. But when learning on graphs or manifolds, since the information now lies in a non-Euclidean domain, the application of known architectures using convolutions, down and up sampling is not well defined. Our work falls within this concept of Geometric Deep Learning that aims to find techniques capable of treating data lying on an unstructured grid.

We present a model that creates a compact representation from 3D deformable shapes that share a common topological structure. The main desired application using this representa-

tion is generation, but others are possible such as classification, segmentation, correspondence or retrieval.

Here, we focus on datasets representing human bodies. The first idea would be to directly treat raw output of scans, but methods having point clouds as input often lack of connectivity information when the studied objects are human shapes that can be deformed with near-isometric transformations. The second idea is to add links to those raw point clouds, transforming the data into triangular meshes with a lot of vertices and changing connectivity, and then feed them to neural networks. As of today, methods are not able to treat efficiently 3D data lying on an unstructured grid with a changing connectivity, especially when talking about human bodies. So most of the time the studied models need a more simplified type of input like meshes with a constant connectivity and a small number of vertices, ie SMPL [1], the human body model which parameterizes the mesh by 3D joint angles and a low dimensional linear shape space.

State of the art methods treat those triangular meshes with constant connectivity either in the spatial domain using spiral convolutions [2] or in the spectral domain using first-order Chebyshev polynomials [3]. In general, these methods are constrained by the number of vertices, making the training expensive and long when meshes have a lot of nodes. Plus, the transfer of known and useful operations such as convolutions, pooling and up sampling from 2D to 3D still remains a challenge. We aim to solve these problems by using spectral methods in a different way.

Inspired by spectral analysis applications, we alleviate the training of a neural network by discarding unessential information contained in very small details. It is known that a signal can be well approximated using a relatively small set of spectral coefficients corresponding to low frequencies. The idea of our work is to take profit from this energy compaction: by feeding a neural network only with coefficients that contain a significant amount of energy, the problems arising when treating triangular meshes like the high number and the non-ordering of vertices can be solved. Indeed, the quantity of data given to the model can be drastically reduced using spectral compaction, and the ordering coming from the difference of magnitudes of these coefficients can be exploited to define an ordering. An illustration of the general process is presented in Figure 1.

Main contributions

We present a process that enables us to use traditional architectures on surfaces by using spectral mesh processing. By transforming the geometry of meshes in the spectral domain with the Graph Laplacian, we obtain spectral coefficients that have a known order. Then, using an Autoencoder like architecture, we can directly apply convolutions and down/up sampling operations to those coefficients. Our method falls within the class of AE-based generative models for 3D shapes. The main contributions of the proposed network are:

- the application of deep learning techniques to spectral coefficients of triangular meshes without going back to the spatial domain,
- an architecture that can treat meshes in an alleviated way by using a smaller number of frequencies than the number of vertices,
- an architecture that gives better results than state of the art methods in a much faster way in order to be able to treat big datasets.

The code will be released so that the generation of the datasets and the training can be reproduced. Also, pre-trained models will be given.

2. Related work

In order to transfer deep learning techniques to 3D data, early methods transform the inputs to voxels or images. Here, we will focus only on methods based on the treatment of surfaces directly. Methods using point clouds, surfaces without connectivity information, only rely on the Cartesian coordinates of each point, whereas graphs/manifolds methods exploit the known connectivity, and can be used in two different domains: spatial or spectral. Interested readers can refer to surveys [4, 5, 6, 7].

2.1. Convolution on point clouds

Most point cloud based methods come from the work of Charles et al. [8] and Qi et al. [9]. Aumentado-Armstrong et al. [10] define a VAE able to disentangle intrinsic and extrinsic information using spectral information prior in an unsupervised way. Cosmo et al. [11] introduced a stronger geometric prior by making the latent space preserve computed geodesic distances using the work from Crane et al. [12] on generated surfaces. Rakotosaona et al. [13] presented a double autoencoder architecture, one extracting features from point clouds and the other one extracting features from edge lengths. Then, by mapping the two latent spaces, the network is able to realistically interpolate between two point clouds using the edge lengths latent space. All these architectures use PointNet as an encoder, enabling to study on variable topology and reducing the limitations in the type of input data employed, but is also making them unable to capture local correlation between neighboor vertices: while having the advantage of using simple and efficient architectures, their capacity to reveal precise local surfaces features are often less than structures having access to connectivity information. More recently, Thomas et al. [14] introduced KPConv that treats each node of a point cloud by weighting it depending on the Euclidean distances with its neighboorhood, thus having more information about the local correlation of the surface. However, those are obtained using a K-nearest algorithm, and wrong local links can be found. Methods using PointNet or KPConv architectures then lack of connectivity information, especially when treating human surfaces that can be deformed using near-isometric transformations, and meshes-like structures are often more performant.

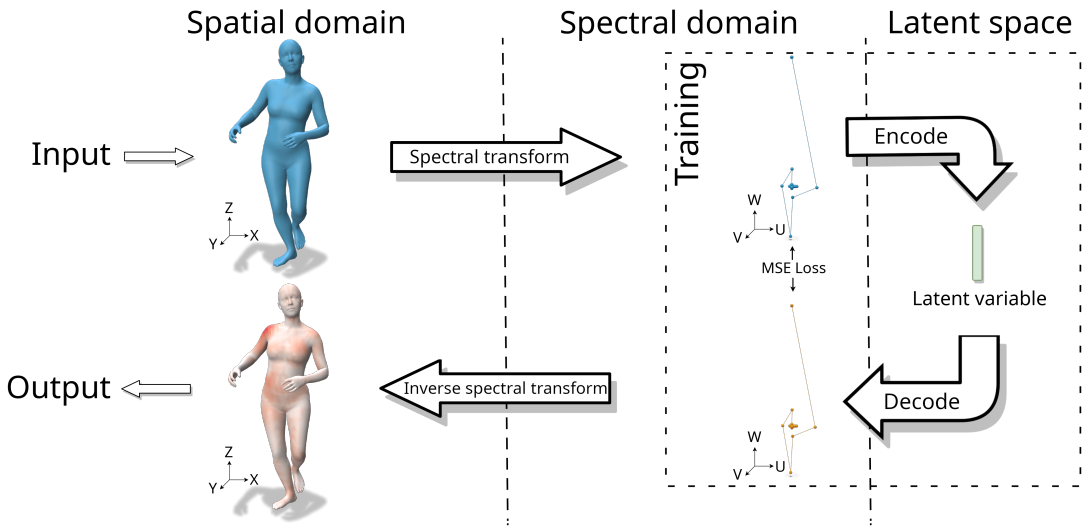

Fig. 1. Illustration of the general process. A mesh, lying in the spatial domain and represented as x, y, z coordinates, is transformed into frequency coefficients lying in the spectral domain represented as u, v, w coordinates. Then, an Autoencoder is trained in order to reconstruct those spectral coefficients by passing them through a bottleneck. There is no return in the spatial domain during training. Finally, the output mesh can be recovered from the output spectral coefficients with an inverse spectral transform.

2.2. Convolution on meshes

The main problem when transferring known deep learning architectures to meshes is that the grid lacks of general structure. Architectures using the connectivity of meshes can be discerned into two categories: the ones using computations in the spatial domain and the ones using computations in the spectral domain.

2.2.1. Spatial domain

In the spatial domain, the global grid in a graph is irregular: there is a need to specify an order for the neighbouring nodes. In order to be able to slide a window kernel over the vertices, spatial based methods need to define convolutions based on these nodes' spatial relations. It is then necessary to compute weights between them. The convolution operation on those irregular graphs has been defined in different ways. First, those weights can be static, learned with a preprocess computation. Technics with mixture models/local parameterization were used: Masci et al. [15] applied filters to local patches represented in geodesic polar coordinates. Boscaini et al. [16] exploited the same idea by formulating local intrinsic patches on meshes, and Fey et al. [17] utilized pre-defined local pseudo-coordinate systems over the graphs. Also, techniques with spiral like convolutions were used: preliminar work by Lim et al. [2] introduced spiral convolutions with SpiralNet. Bouritsas et al. [18] used similar spiral convolutions with Neural3DMM coupled with an Autoencoder based on CoMA architecture [19]. Gong et al. [20] proposed an upgraded version with SpiralNet++ [20]. Then, those filters can be dynamically learned. Monti et al. [21] introduced MoNet, a mixture model with learned weights. Verma et al. [22] presented FeastNet, a graph convolution operator enabling the computation of dynamic correspondences between kernel weights and neighboring nodes with arbitrary connectivity. Zhou et al. [23], inspired by the spiral method, proposed vertex-wise weighted convolutions.

The advantage of spatial techniques is that they generalize across domains and are able to learn filters that are intrinsic and accurate. The downside is that they are not robust to near-isometric deformations and can be time consuming if meshes have a lot of vertices.

2.2.2. Spectral domain

The concept of the spectral domain methods rely on the convolution theorem saying that a convolution in the spatial domain is equivalent to a pointwise product in the spectral domain. Bruna et al. [24] first used the Laplacian eigenvectors to project features/kernel on them and to multiply those projections before going back to the spatial domain, but this resulted in a slow process. Instead of computing eigenvectors, Defferrard et al. [3] used truncated Chebyshev polynomials and Kipf et al. [25] used only first-order Chebyshev polynomials, resulting in a faster way to do convolutions in the spectral domain. Ferrari et al. [19] used Convolutional Mesh Autoencoder (CoMA) based on ChebyNet and spatial pooling. They generalized downsampling and upsampling layers to meshes by collapsing unimportant edges based on quadric error measure, that some spatial methods also use.

A drawback of these methods is that they still use spatial downsampling/upsampling. This prevents to fully exploit the profits of the spectral domain: the speed of computation. Also, more recent spatial methods outperformed their ability to reconstruct and generate meshes from a latent space. We aim to use spectral mesh analysis in a different and novel way.

The challenge when trying to develop convolutions on graphs or manifolds is the lack of an order for the nodes. Also, the number of vertices of meshes in a dataset is still a constraint. To overcome those problems, we aim at designing a model that works on an ordered point cloud: the spectral coefficients. This enables the direct application of convolutions without needing

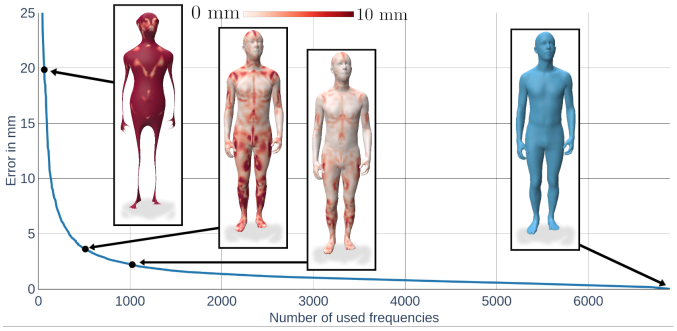

Fig. 2. Transform / inverse transform mean errors as a function of the chosen number of eigenvectors. Displayed meshes correspond to 64, 512, 1024 and 6890 frequencies respectively. By using all available frequencies (6890, the number of vertices), the exact geometry is recovered.

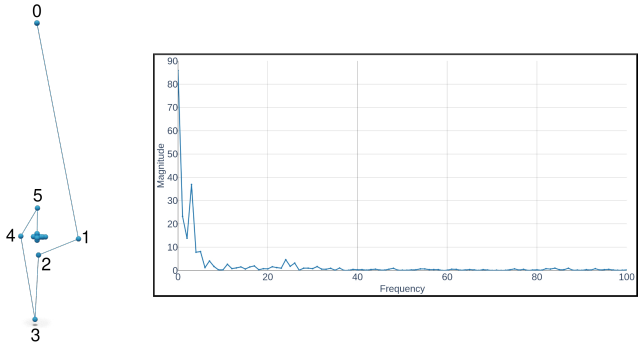

Fig. 3. On the left, example of low frequency spectral coefficients in 3D (only the 6 first are labeled for visibility). On the right, magnitudes of the first 100 coefficients. Low frequencies have higher magnitudes than high frequencies.

to define a special ordering. Also, our model does not need to go back to the spatial domain since the down/up sampling operations are done in the spectral domain. This allows us to develop a process that is fast and that only needs for preprocessing the computation of eigenvectors of one mesh from a dataset with constant connectivity and the spectral transformation of all meshes in the dataset to spectral coefficients.

We start by reminding the reader about spectral mesh analysis, allowing the transformation of a triangulated surface to the frequency domain. Then, we introduce the Spectral Autoencoder, a neural network that takes as input spectral coefficients containing compacted meshes' geometry information.

3. Reminder about spectral mesh processing

In spectral mesh processing, surfaces are studied through operators which are usually variants of the Laplacian and provide new bases serving various processing applications. Vallet et al. [26] proposed a fast computation algorithm for the Laplace-Beltrami eigenfunctions of meshes up to a million vertices. Reuter et al. [27] introduced a method to extract Shape-DNA, a signature made of the Laplace–Beltrami operator's eigenvalues. Here, we will focus on the use of the topological Laplacian [28].

A triangular mesh M can be expressed as a set of 3D points P coupled with a triangulation. Each point $p_i \in P$ is represented as absolute Cartesian coordinates: $p_i = (x_i, y_i, z_i), i \in [1, n]$, n being the number of vertices. It is then possible to transform those absolute coordinates to spectral coefficients by using a topological operator: the Graph Laplacian [28]. If A is the adjacency matrix, and D is the diagonal matrix of degrees of each vertex, then the Graph Laplacian L is defined as:

$$L = D - A \tag{1}$$

It is important to note that a dataset of meshes with a constant connectivity will have only one common Graph Laplacian.

This Graph Laplacian is a square matrix of size $n * n$, and can be decomposed into a chosen number of k scalar eigenvalues λ_i and k eigenvectors ϕ^i with $i \in [1, k]$ and $k <= n$. Pairs of eigenvalue and eigenvector satisfy the equation $-L\phi^i = \lambda_i \phi^i$.

The eigenvectors are of size n and correspond to columns of the matrix:

$$\Phi = \begin{pmatrix} \phi_1^1 & \phi_1^2 & \dots & \phi_1^k \\ \vdots & \vdots & \vdots & \vdots \\ \phi_n^1 & \phi_n^2 & \dots & \phi_n^k \end{pmatrix} \tag{2}$$

Using these eigenvectors, it is possible to transform the absolute coordinates of mesh vertices to spectral coefficients and to inverse transform the spectral coefficients to absolute coordinates using matrix multiplications: $C = \Phi^T \cdot P$ and $P = \Phi \cdot C$ respectively. Depending on the chosen number k of eigenvectors, the recovered geometry after an inverse transform will be more or less low pass filtered (see Figure 2).

Spectral coefficients C are a set of 3D points. Each point can be represented as frequency coordinates: $c_i = (u_i, v_i, w_i), i \in [1, k]$ (see Figure 3). They contain information about the geometry of the original vertices in a compressed form. More specifically, the most important part is contained in the low frequencies, representing the general aspect of the surface, whereas the details are located in high frequencies. The idea of our work relies on this fact: visually, small details in high frequencies could be discarded, enabling to treat less information with approximately the same precision. This is why the truncated spectral coefficients are the input of the presented neural network in the next section.

Spectral mesh processing therefore provides a basis defined by the eigenvectors for mesh compression/reconstruction. The spectral transform and the inverse spectral transform being linear functions, operations in the frequency domain (e.g. interpolation) exhibits the same artifacts as in the spatial one like non conservation of edge length or triangle area. In early works, the frequency domain was used to alleviate the task of editing or morphing often done in the spatial domain by truncating the number of used frequencies. Here, we aim at alleviating the work of a deep learning model by directly giving as input truncated spectral coefficients in order to extract useful features in

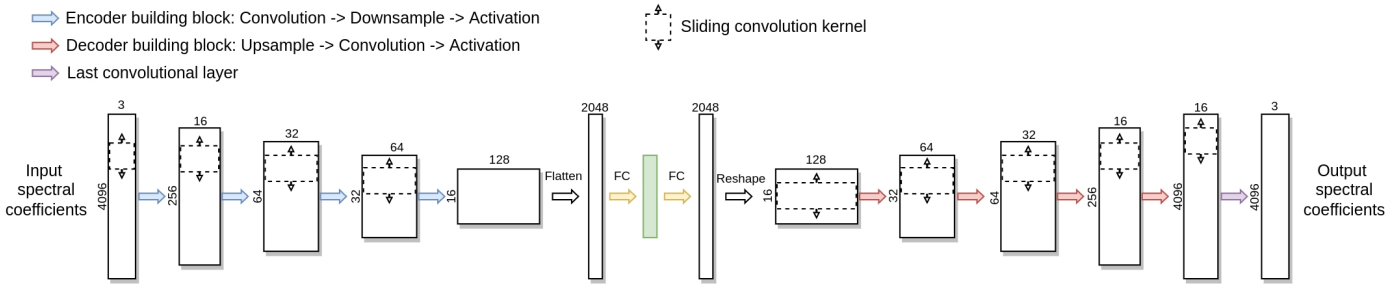

Fig. 4. Illustration of the Spectral Autoencoder with learned pooling using 4096 frequencies (SAE-LP-4096).

an unsupervised way. This allows to go further and create an even more compact latent space.

4. Spectral Autoencoder

We now introduce how our Spectral Autoencoder (SAE) takes the spectral coefficients as input. The general process is presented in Figure 1 and an illustration of the Spectral Autoencoder is shown in Figure 4. First, the geometry of a mesh is transformed to spectral coefficients using the spectral transform presented in the previous section. Then, those spectral coefficients are given to a deep convolutional Autoencoder able to learn hierarchical representations by making the input go through multiple layers. We here introduce how the data is preprocessed and how the convolutions and down/up sampling operations are applied to the spectral coefficients.

4.1. Preprocessing

The first step of preprocessing is the computation of the eigenvectors. The idea here is to not use all available frequencies, but rather a truncated version of them. In section 5, we show the impact of using different number of frequencies. Then the vertices of all samples in the dataset are transformed into spectral coefficients. We can see in Table 1 the preprocessing times of computation in terms of number of eigenvectors. They are reasonable in comparison with the spirals time calculation from [18, 20].

4.2. Convolutions on spectral coefficients

Convolutions are the major building blocks in deep learning applications. In 2D, they allow the extraction of useful features from images in an efficient way since they are fast to compute and they reduce the number of parameters of a neural network. It is therefore natural to try to extend the application of convolutions to other domains than images. As said in Section 2, the main contributions to apply convolutions on triangular meshes take place either in the spectral domain or in the spatial domain. Ranjan et al. [19] built a convolutional autoencoder (CoMA) upon ChebNet with spectral convolutional filters, resulting in isotropic kernels with limited expressiveness. Bouritsas et al. [18] improved those results with a spiral convolution operator (Neural3DMM [18]) that defines an explicit order of the neighbors, resulting in anisotropic filters. However, this method requires to define starting vertices for spiral orders, preventing from efficiently exploiting the irregular structure of the connectivity. Additionnaly, performance decays are involved since zero-padding is needed in order to have fixed-size spirals. Gong et al. [20] introduced SpiralNet++, an improved version of Neural3DMM, making the convolutions faster by avoiding the zero paddings.

SpiralNet++, which is the most efficient and fastest way to do convolutions on meshes, still needs to define an order for convolutions on vertices, making the implementation a complex one. In our work, we simplify this process by doing convolutions on the array of spectral coefficients in a natural way since the frequencies are already ordered. Figure 4 shows an illustration of the presented Spectral Autoencoder with learned pooling using 4096 frequencies (SAE-LP-4096). By simply sliding a convolution kernel over the coefficients, we show that the network is able to learn interesting features.

4.3. Pooling

The behaviour of a neural network is closely related to the pooling procedure. Classical works for 1D signals or 2D images use a sliding window in order to retain only the maximum values in local regions for downsampling, and add values for upsampling to bring back a higher resolution for the next layer. For meshes, Ranjan et al. [19] introduced a down/up sampling method in the spatial domain. Downsampled meshes are computed by contracting edges while maintaining surface error approximation and upsampled meshes are computed creating vertices from the triangles barycentric coordinates of the downsampled meshes. Those operations are represented as transform matrices. Some works proposed to learn these aggregation weights with dense mapping [29, 30] or fully-connected layers [31]. Chen et al. [32] introduced a method where they are learned through an attention module in order to avoid over-parameterization.

In our work, we chose to down/up sample the spectral coefficients with two methods. The first one consists in applying classical maxpooling/upsampling operations. We will refer to this method as **Spectral Autoencoder - Classic Pooling** (**SAE-CP**). The second one consists in learning the mapping matrices, as in the work from Chen et al. [32]. But we did not use an attention module since learning directly parameters is simpler and we do not use all available frequencies so it does not lead to over-parameterization. We will refer to this method as **Spectral Autoencoder - Learned Pooling** (**SAE-LP**). In this case,

down/upsampling matrices are simply filled with learnable parameters. Then, during training, these parameters are learned along with the other parameters of the model. See Figure 5.

We show in the next section the comparison of using classical maxpooling/upsampling and using learned down/up sampling matrices, along with a comparison between multiple choices of used number of frequencies.

5. Evaluation

In this section, we evaluate the presented models. We first introduce used datasets and give details on the implementation. Then, we evaluate our best model against two baselines: Neural3DMM [18] and SpiralNet++ [20] by presenting quantitative and qualitative results and by comparing the speed of computation. Next, an ablation study shows that using less frequencies still gives good result while reducing the number of parameters of models. Then, we show that it is possible to interpolate in the latent space in order to generate realistic meshes.

5.1. Datasets

DFAUST. The dataset from the work of Bogo et al. [33] consists of 41,220 body meshes having 6890 vertices from 10 identities performing multiple actions. We split the data into 32,535 samples for training representing the first 8 identities and 8,685 samples for testing representing the last 2 identities.

AMASS. The dataset from the work of Mahmood et al. [34] is a unification of 15 smaller ones by fitting SMPL [1] body model to motion capture markers, consisting of 344 subjects and more than 10K motions. We follow the protocol splits: 1 out of 100 frames is selected for the middle 90% portion of each sequence, resulting in 111,327 meshes for training and 10,733 for testing. Identities are not shared between the train and test dataset. We preprocess the surfaces so that they are centered at the origin and oriented towards the same direction.

The metric used for the experiment is a measure of the quality of reconstructed meshes from the latent space. It is computed as the average distance in millimeters between corresponding vertices of the input and output meshes. This measures the capacity of the model to obtain a compact representation and to generalise to novel surfaces from the distribution it was trained on. All models are not normalized and have the actual size of the person.

5.2. Implementation

We follow the setting of previous works for the architectures of models.

Neural3DMM and **SpiralNet++**: the convolutional filters of the encoder have sizes [3, 16, 32, 64, 128]. A fully-connected layer then maps the data to the wanted latent size. After another fully-connected layer, the convolutional filters of the decoder have sizes [128, 64, 32, 16, 16]. A last convolutional layer maps the data to the number of features of the geometry, 3. Dilated convolutions with h = 2 hops and dilation ratio r = 2 are used for the first and the last two layers of the encoder and the decoder

Method	Eigenvectors	DFAUST	Total DFAUST	AMASS	Total AMASS
Neural3DMM	-	-	~ 30s	-	~ 30s
SpiralNet++	-	-	~ 30s	-	~ 30s
SAE-*-6890	~40s	~3s	~ 43s	~16s	~ 56s
SAE-*-4096	~28s	~2s	~ 30s	~12s	~ 40s
SAE-*-2048	~36s	~1s	~ 37s	~8s	~ 44s
SAE-*-1024	~9s	~0.5s	~ 9.5s	~7s	~ 16s
SAE-*-512	~3.5s	~0.3s	~ 3.8s	~3.2s	~ 6.7s

Table 1. Comparison of preprocessing times. For Neural3DMM and SpiralNet++, spirals are computed only once on a template mesh, so the time does not depend on the dataset size. SAE-*-k stands for our Spectral Autoencoder with classic pooling or learned pooling using k frequencies. For our method, this time depends on the eigenvectors computation (column Eigenvectors) and on the transformation of all meshes in a dataset to spectral coefficients (columns DFAUST and AMASS). Even with 4096 frequencies, the preprocessing time is reasonable compared to spirals method. The fact that the computation of eigenvectors for 2048 frequencies is longer than for 4096 comes from the eigensolver.

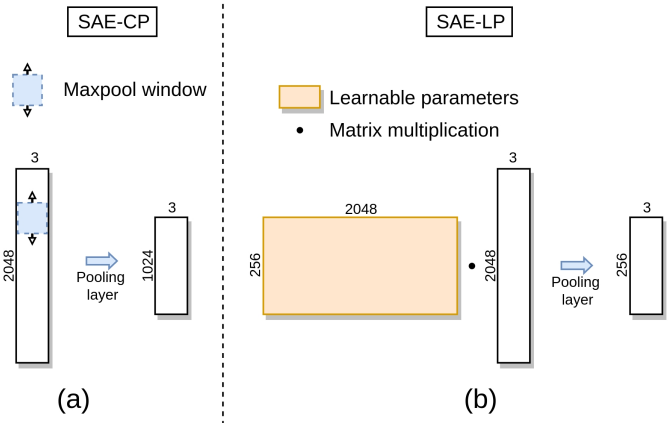

Fig. 5. Illustration of the two pooling methods used when downsampling 2048 frequencies. (a) - A classic pooling window of size 2 with stride 2 allows to reduce the resolution of the spectral coeffcients by a factor of 2. This method is referred to Spectral Autoencoder with classic pooling (SAE-CP). (b) - the downsampled spectral coefficients are computed by multiplying them with a matrix containing learnable parameters. The downsampling factor is larger than with the classic pooling (see Section 5.2 for more information). This method is referred to Spectral Autoencoder with learned pooling (SAE-LP).

respectively. The sizes of the spirals are [12, 14, 9, 9] for the encoder and [9, 9, 14, 12, 12] for the decoder.

We recall that for the two baselines, the inputs are the meshes' Cartesian coordinates from the spatial domain standardized with mean equal to zero and standard deviation equal to one. For our models, the inputs are the spectral coefficients.

SAE-CP-*k*: the construction of the **S**pectral **A**uto**e**ncoder with **c**lassic **p**ooling using *k* frequencies follows common Autoencoder architectures using convolutions and pooling/upsampling operations. The chosen number of frequencies of the input spectral coefficients is always a power of two. Then, the number of layers of the encoder depends on the number of times we have to divide by two in order to have 32 remaining frequencies after the last maxpooling step: 4 steps for 512 frequencies, 5 steps for 1024 and so on. For 512 frequencies, the convolutional filters of the encoder have sizes [3, 32, 64, 64, 128]. The decoder has convolutional filters of sizes [128, 64,

64, 32, 32, 3]. If more layers are needed because more frequencies are used, we duplicate the filters of size 64. The sizes of the convolutions kernels are 3 for all layers with a padding of 1 so that the length of the input is not modified after the convolution is applied. Also, the size of the window for maxpooling and the scale factor of the upsampling is 2 for all layers.

SAE-LP-*k*: the construction of the **S**pectral **A**uto**e**ncoder with **l**earned **p**ooling using k frequencies is similar to the previous one except for the down/up sampling layers. The convolutional filters of the encoder have sizes [3, 16, 32, 64, 128]. Instead of doing classical maxpooling or upsampling, mapping matrices containing learnable parameters are created. Then, those parameters are learned along with the other components of the model. The sizes of the matrices for the encoder are [k, 256, 64, 32, 16], k being the chosen number of frequencies. For the decoder, the convolutional filters have sizes [128, 64, 32, 16, 16, 3]. The matrices for upsampling are of the same size as the encoder ones in reverse order. The sizes for the convolutions kernels are also 3 for this model with padding of 1.

All models are trained on the same hardware. The batch size is 16, the learning rate is 1e-4 and a scheduler is used so that the learning rate is reduced by a factor of 0.1 when the reconstruction has stopped improving (with a threshold of 1e-4) for 3 epochs. Models are trained for a maximum of 20 hours.

5.3. Comparison with baselines

We first evaluate our model giving the best results: the SAE-LP-4096. Differences between mean reconstructions on test datasets, visual quality of the reconstructed meshes and the times per epoch are compared to the two baselines Neural3DMM [18] and SpiralNet++ [20].

5.3.1. Quantitative results of reconstruction

We follow [18] for the choice of latent sizes, based on the variance explained by PCA of roughly 85%, 95% and 99% of the total variance. Fig 6 shows results of reconstruction accuracy on the DFAUST and AMASS dataset. Here, the SAE-LP-4096 is a model that takes as input 4096 frequencies. We can see that for all latent sizes, our model outperforms the two baselines. Table 2 shows the number of parameters of the three neural networks. The SAE-LP-4096 clearly has a lot more, but 90% of these parameters are concentrated in the first downsampling and the last upsampling matrix (see Section 5.2). Additionnaly, we show in the Section 5.4 that models with the same architecture but without that much parameters still manage to get competitive results.

Moreover, we can compare the compression capacity of the model's latent space with the compression capacity of the spectral domain. This can be done by simply measuring the reconstruction error after applying a spectral transform and then an inverse spectral transform (as in Fig 2) on all the test dataset's meshes when using a number of frequencies similar to the evaluated number of latent dimensions. On the DFAUST dataset, the mean reconstruction errors when using 3, 6 and 22 frequencies (resulting in 9-18-66 dimensions respectively since there are 3 coordinates u, v, w) are 368.1±42.7, 96.5±10.7 and 54.7±3.6 millimeters respectively. Fig 6 exhibits reconstruction errors for 8-16-64 latent dimensions of 55.5±16.9, 33.0±10.7, 10.3±2.7 millimeters respectively. This clearly shows that the latent spaces built by the model have a better compression capacity.

Method	Latent size 8	16	64
Neural3DMM	274K	331K	675K
SpiralNet++	415K	471K	802K
SAE-LP-4096	2.23M	2.26M	2.46M

Table 2. Comparison of the number of parameters in function of the latent space size. In this case, our model uses 4096 frequencies. More than 90% of the parameters for the SAE-LP-4096 are located in the first downsampling and the last upsampling matrices.

5.3.2. Qualitative results of reconstruction

Visual reconstructions are shown in Fig 7. We compare models all with a latent dimension of 64. The main observation is that when the baselines have to handle parts of bodies in a position not oftenly saw during training (especially arms and hands), we can see that the details are more degenerated. In contrast, the Spectral Autoencoder is able to reconstruct them with more smoothness, leading to visually better results. This can be explained by the fact that, during early training, our models first learn to better reconstruct the information contained in the low frequencies since low coefficients have a higher magnitudes than the high frequency ones. This leads to body parts in the right position but without enough details. Then, in late part of training, the model learns to reconstruct the details. In contrast, the baselines struggle to reconstruct those body parts.

5.3.3. Time per epoch

The main advantage of our method is the speed of computation. We can see in Table 3 the difference of time per epoch for Neural3DMM, SpiralNet++ and the SAE-LP-4096. Since we do not take as input all available frequencies, the process is much faster, while still having access to important frequencies. Also, Neural3DMM and SpiralNet++ need to rearange the arrays of vertices in order to do convolutions specified by the precomputed spirals, unlike our method where convolutions are done on arrays in a classical way. Then, even if SpiralNet++ managed to decrease the computation time per epoch compared to Neural3DMM, our network is way faster.

We showed that the Spectral Autoencoder with learned pooling using 4096 frequencies is able to learn features on triangular meshes and construct a latent space where reconstruction is possible, giving better results than state of the art methods. We now show the impact of using different configurations for our architecture.

5.4. Ablation study

In this section, we evaluate the behaviour of our architecture using learned pooling, classic pooling and a different number of frequencies. We first present quantitative results in order to

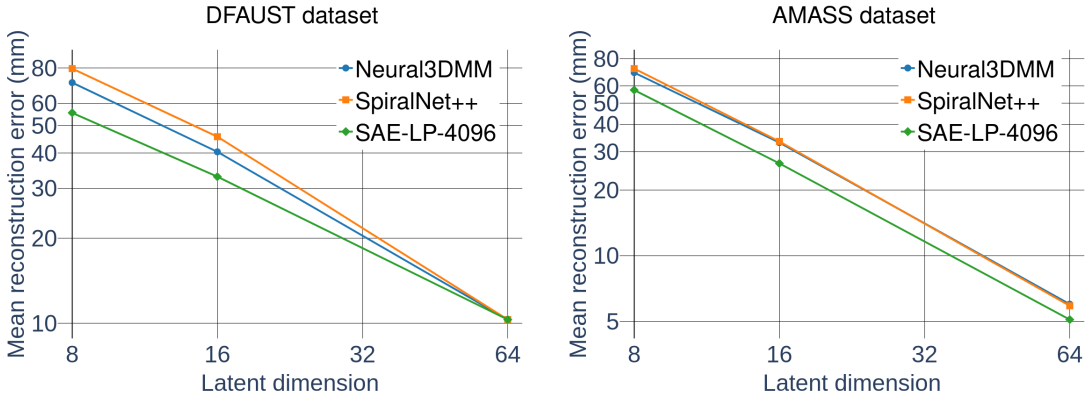

Fig. 6. Reconstruction results on DFAUST and AMASS datasets between Neural3DMM, SpiralNet++ and our best model. For all latent sizes, our method outperforms the two baselines.

Method	DFAUST	AMASS
Neural3DMM	~ 156s	~ 472s
SpiralNet++	~ 68s	~ 200s
SAE-LP-4096	~ **28s**	~ **83s**

Table 3. Time per epoch comparison on DFAUST and AMASS Datasets between the SAE-LP-4096 and the baselines. Our model is faster by a large margin.

Method	AMASS
SAE-CP-512	~ **84s**
SAE-CP-1024	~ 89s
SAE-CP-2048	~ 93s
SAE-CP-4096	~ 115s

Table 4. Time per epoch of models with classic pooling. For the SAE-CP, increasing the number of given frequencies leads to higher computation times because the factor of downsampling is 2, leading to arrays of bigger sizes in hidden layers. See Figure 5.

.

compare the mean reconstruction error with different configurations. Then, results are presented showing the capacity of a model trained on a dataset to generalise on the other dataset. Finally, qualitative results show that the reconstructed meshes are still visually acceptable even when using less frequencies.

5.4.1. Quantitative results

Tables 7 and 8 show the number of parameters in function of the number of used frequencies and the size of the latent space. Results of reconstruction on the AMASS dataset using different configurations are presented in Figure 8. The red line indicates the best attainable score for each number of used frequencies, corresponding to the mean error on the test dataset after an inverse spectral transform, see Figure 2. It is clear that when using 512 frequencies, our models can't have a better score for a latent dimension of 128. First, for a latent dimension of 16, both our architectures give better results. For a latent dimension of 32, the SAE-CP gives similar results as the baseline, while the SAE-LP still has a better reconstruction score. For higher

Method	AMASS
SAE-LP-512	~ **80s**
SAE-LP-1024	~ 83s
SAE-LP-2048	~ 83s
SAE-LP-4096	~ 83s

Table 5. Time per epoch of models with learned pooling in function of used frequencies. Even with more frequencies, the time of computation remains constant.

latent dimensions, the SAE-CP struggles and outputs less accurate meshes while the SAE-LP gives similar results than the baselines when using a low number frequencies, and gives better scores when using more frequencies. The worse behaviour of the SAE-CP comes from the pooling method, corrected by the one used with the SAE-LP. Also, when using too few frequencies, the networks do not have access to details information, leading to meshes without enough precision. Additional results are given in the table in supplementary material.

5.4.2. Qualitative results

Different levels of details on a head and a foot of reconstructed meshes with different number of used frequencies are presented in Figure 9. We can see that by giving enough frequencies to the model, it is able to reconstruct meshes with as much precision as Neural3DMM. Then, Figure 10 shows visual results compared to SpiralNet++. For reconstructed meshes with 1024 frequencies, we can see that details contained in high spectral coefficients are missing: this results in symmetric parts on the body with small errors (see Figure 2 for a comparison). Nevertheless, the model using only 1024 frequencies still manages to reconstruct some parts of the body with more precision and more smoothness compared to SpiralNet++. Then, the models using more frequencies give the same kind of results as the one with 1024 frequencies but with more details.

5.4.3. Time per epoch

Times per epoch for the SAE-CP are first presented in Table 4. We can see that when treating more frequencies, the network gets slower but is still faster than SpiralNet++. This is

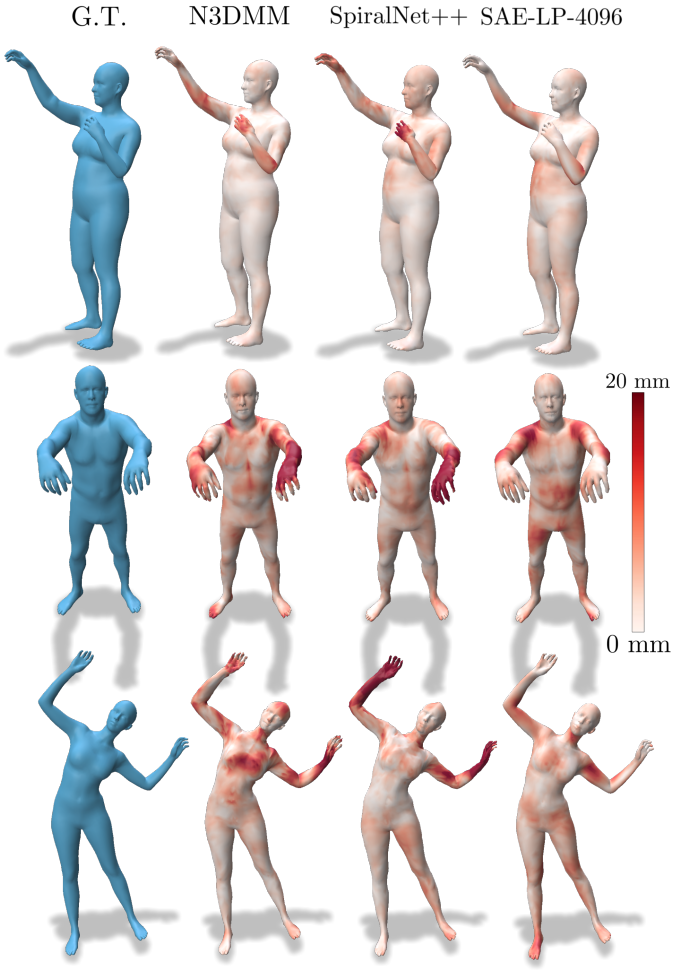

Fig. 7. Comparison of reconstructions between Neural3DMM (left), SpiralNet++ (middle) and our model using 4096 frequencies, the SAE-LP-4096 (right). The latent dimension is 64. When reconstructing parts of body not usually seen in the dataset, our model produces more detailed or more smooth surfaces, especially on the hands.

because the SAE-CP number of layers depend on the number of frequencies as input, leading to more convolutions applied to bigger arrays of spectral coefficients. Then, Table 5 shows the computation time per epoch of the SAE-LP. Taking as input more spectral coefficients does not deteriorate the speed and still gives training times that are much shorter than baselines. This is probably the main advantage of our method since training on datasets with a lot more samples, like the AMASS dataset sampled with more frames, is now feasible in a reduced time.

5.4.4. *Crossing databases*

We then try to observe the ability of a model trained on a dataset to generalise on a different one. Table 6 shows results when crossing datasets. Naturally, models trained on the DFAUST dataset struggle to reconstruct meshes from the AMASS dataset since the latter is larger. Inversely, models trained on the AMASS dataset manage to get a score that is close to the one obtained by models trained on the DFAUST dataset. This shows the advantage for a model of being fast, thus able to learn on a large dataset.

Latent size	Test dataset	Train dataset	Mean (mm)	Cross error (mm)
8	DFAUST	DFAUST	55.5	-
		AMASS	64.0	**+8.5**
	AMASS	AMASS	57.4	-
		DFAUST	119.2	+61.8
16	DFAUST	DFAUST	33.0	-
		AMASS	35.9	**+2.9**
	AMASS	AMASS	26.5	-
		DFAUST	73.6	+47.1
64	DFAUST	DFAUST	10.3	-
		AMASS	17.9	**+7.6**
	AMASS	AMASS	5.1	-
		DFAUST	27.9	+22.8

Table 6. Comparison of reconstruction errors when crossing datasets using the model SAE-LP-4096. The cross error is the difference of mean reconstruction between models trained on different datasets but evaluated on the same. A model trained on AMASS, a big dataset, is able to correctly reconstruct the DFAUST dataset which is smaller.

5.5. *Interpolation*

Finally, we show the interpolation capacity of our model compared to Neural3DMM. By performing linear algebra in the latent space, it is possible with an Auoencoder-like architecture to generate new samples. We select two different samples from the test set c_1 and c_2, encode them to their latent representations z_1 and z_2, and produce new meshes by sampling along the line: $z = a * z_1 + (1 - a) * z_2$, $a \in [0, 1]$. Figure 11 compares interpolations between Neural3DMM and the SAE-LP-4096. For the first line, meshes on the extremities are the ones sampled from the test dataset, and the three meshes in the middle correspond to linear interpolation of the Cartesian coordinates. When using all frequencies, the result is the same when interpolating spectral coefficients since the Laplacian transform is a linear function. For the two other lines, meshes on the extremities are the reconstructions of the two sampled meshes. Then, interpolated and decoded latent codes are presented in the three middle columns for different a values. For the interpolation in the spatial domain, the arms' length of the model are greatly reduced, which is typically an unwanted behaviour when interpolating human bodies. The two models manage to overcome this problem, meaning that they built a latent space representing the manifold of possible poses for a human body. Nevertheless, as seen previously, the reconstruction quality of our model is better than the Neural3DMM one, especially for a latent dimension of 16 used here, leading to cleaner generated meshes. Also, it is important to note that for all models, the interpolation capacity is well represented only for this size of latent space where both mean reconstruction error and latent space size are not too high.

6. Conclusion

This work falls in the category of Geometric Deep Learning applied to triangular human meshes. The problem with current state of the art methods is that they are limited by the high number of vertices, the unstructured grid and the time of computation. We showed in this paper that by using spectral mesh

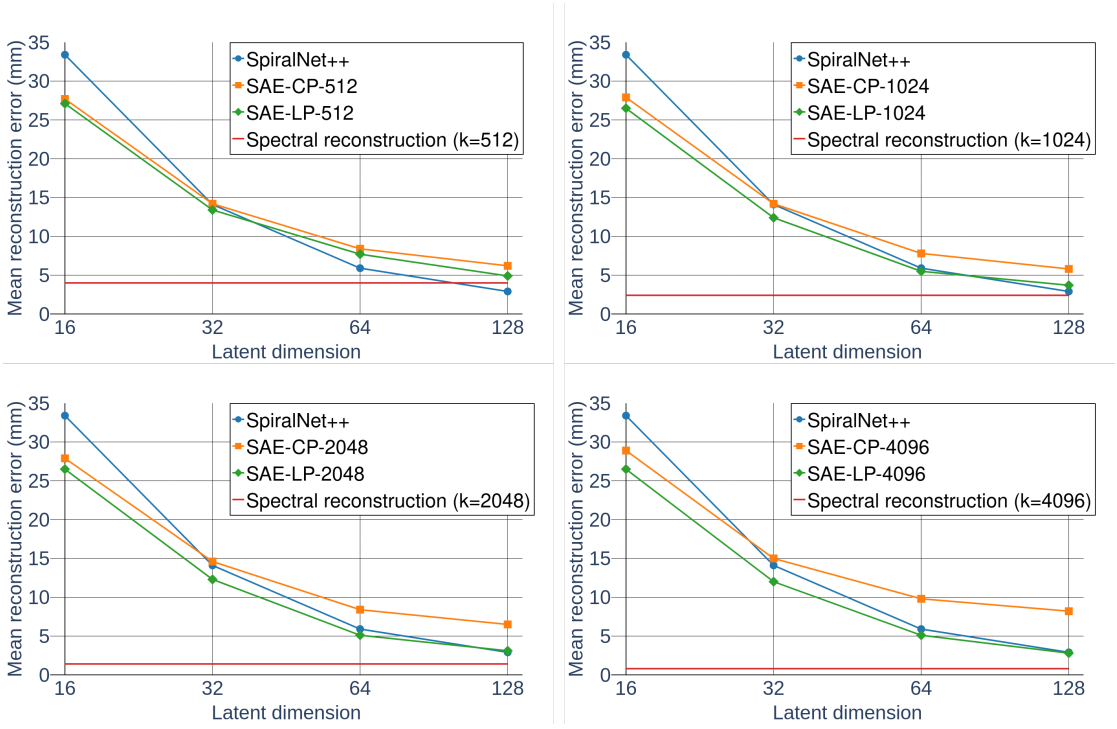

Fig. 8. Comparison of reconstruction results between the baseline SpiralNet++, the SAE-CP and the SAE-LP using different number of frequencies and latent dimensions on the AMASS dataset. The minimum attainable, represented as *Spectral reconstruction*, is computed for each number of frequencies and corresponds to the mean error on the test dataset between original meshes and reconstructed meshes after an inverse spectral transform. Even when using a simpler pooling method or less frequencies, the SAE is able to give competitive results.

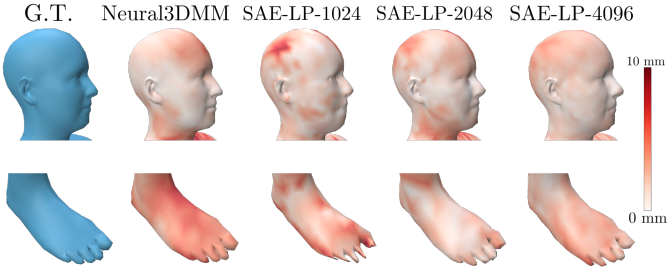

Fig. 9. Examples of details on reconstructed meshes with models using different number of frequencies, zoomed on head and foot. Giving the model access to higher frequencies leads to reconstructed meshes with more details.

processing methods on triangulated surfaces, we are able to solve multiple problems. First, directly treating frequency coefficients instead of Euclidean ones allows the direct application of convolutions since spectral coefficients are ordered depending on their magnitudes. This notion also allows to increase the speed of the networks since convolutions are done on arrays without rearrangement instead of state of the art methods. Furthermore, it is possible to increase the number of vertices of meshes from the dataset while keeping the same neural network architecture since there is no computation in the spatial domain. Finally, upon these advantages, we showed that our models give better results than state of the art methods in terms of reconstruction and interpolation.

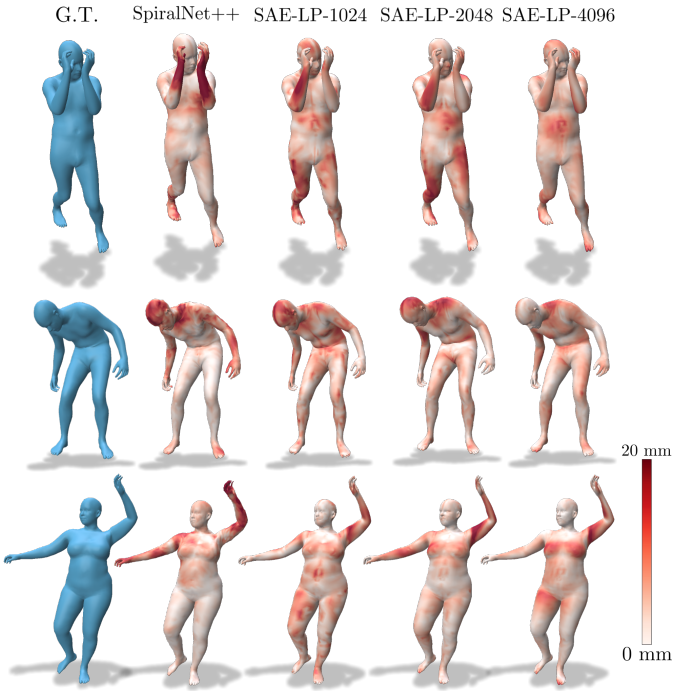

Fig. 10. Comparison of reconstructions between SpiralNet++ and the Spectral Autoencoder with learned pooling using 1024, 2048 and 4096 frequencies. The latent dimension is 64 for all models. While showing errors corresponding to the lack of high frequency information, the model using 1024 frequencies still manages to reconstruct parts of the body in a better way than SpiralNet++.

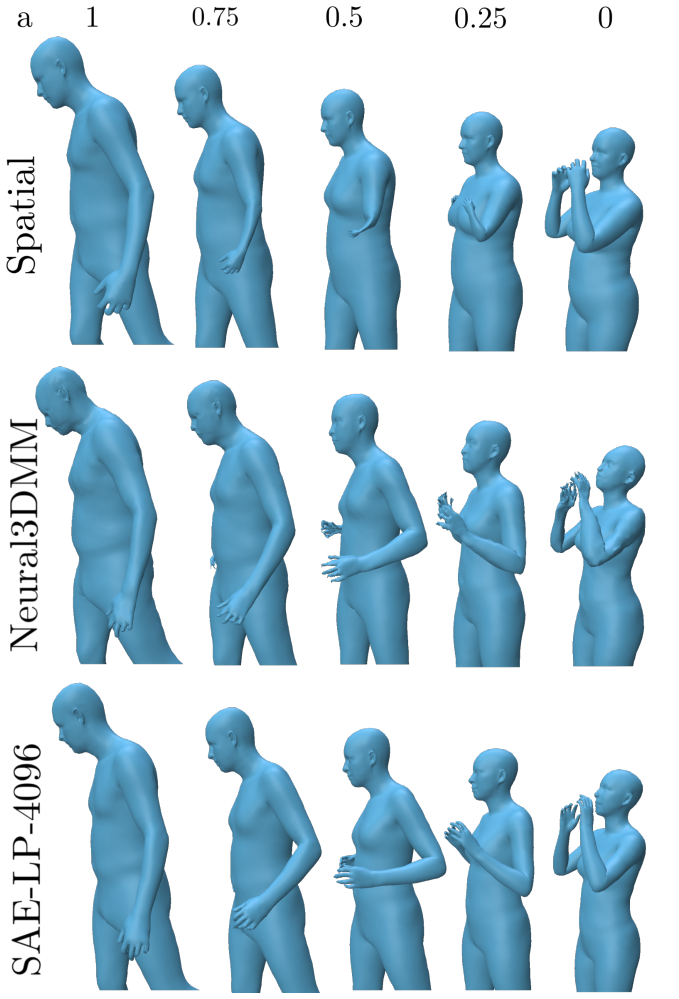

Fig. 11. The first line shows a linear interpolation of the two meshes' Cartesian coordinates on the right and on the left. The second and third line shows mesh generation by interpolating two samples' latent codes in the latent space. The latent dimension is 16 for the two models.

While our method still needs a constant connectivity (as state of the art ones), it is possible to synchronize bases computed from different triangulation. Future works will then be focused on trying to generalize this process to shapes of arbitrary topologies, enabling to directly work on triangulated output from raw scans with a high number of vertices and a changing connectivity while keeping the same number of parameters and the same speed of computation.

Acknowledgments

This work was supported by the ANR project Human4D ANR-19-CE23-0020.

Method	Latent size 8	16	32	64	128
SAE-LP-512	400K	433K	499KK	630K	892K
SAE-LP-1024	662K	695K	761K	892K	1.15M
SAE-LP-2048	1.18M	1.22M	1.28M	1.41M	1.67M
SAE-LP-4096	2.23M	2.26M	2.33M	2.46M	2.72M

Table 7. Number of parameters for Spectral Autoencoders using learned pooling. Most of the parameters are contained in the first downsampling and the last upsampling matrices, especially when the number of used frequencies is high.

Method	Latent size 8	16	32	64	128
SAE-CP-512	159K	225K	356K	618K	1.14M
SAE-CP-1024	184K	250K	381K	643K	1.16M
SAE-CP-2048	209K	274K	405K	668K	1.19M
SAE-CP-4096	233K	299K	430K	692K	1.21M

Table 8. Number of parameters for Spectral Autoencoders using classic pooling.

References

[1] Loper, M, Mahmood, N, Romero, J, Pons-Moll, G, Black, MJ. SMPL: a skinned multi-person linear model. ACM Transactions on Graphics 2015;34(6):1–16. URL: https://dl.acm.org/doi/10.1145/2816795.2818013. doi:10.1145/2816795.2818013.

[2] Lim, I, Dielen, A, Campen, M, Kobbelt, L. A Simple Approach to Intrinsic Correspondence Learning on Unstructured 3D Meshes. arXiv:180906664 [cs] 2018;URL: http://arxiv.org/abs/1809.06664; arXiv: 1809.06664.

[3] Defferrard, M, Bresson, X, Vandergheynst, P. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. arXiv:160609375 [cs, stat] 2017;URL: http://arxiv.org/abs/1606.09375; arXiv: 1606.09375.

[4] Xu, K, Kim, VG, Huang, Q, Kalogerakis, E. Data-Driven Shape Analysis and Processing. arXiv:150206686 [cs] 2015;URL: http://arxiv.org/abs/1502.06686; arXiv: 1502.06686.

[5] Bronstein, MM, Bruna, J, LeCun, Y, Szlam, A, Vandergheynst, P. Geometric deep learning: Going beyond euclidean data. IEEE Signal Processing Magazine 2017;34(4):18–42. URL: https://doi.org/10.1109/msp.2017.2693418. doi:10.1109/msp.2017.2693418.

[6] Xiao, YP, Lai, YK, Zhang, FL, Li, C, Gao, L. A survey on deep geometry learning: From a representation perspective. Computational Visual Media 2020;6(2):113–133. URL: https://link.springer.com/10.1007/s41095-020-0174-8. doi:10.1007/s41095-020-0174-8.

[7] Wu, Z, Pan, S, Chen, F, Long, G, Zhang, C, Yu, PS. A Comprehensive Survey on Graph Neural Networks. IEEE Transactions on Neural Networks and Learning Systems 2021;32(1):4–24. URL: https://ieeexplore.ieee.org/document/9046288/. doi:10.1109/TNNLS.2020.2978386.

[8] Charles, RQ, Su, H, Kaichun, M, Guibas, LJ. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI: IEEE. ISBN 978-1-5386-0457-1; 2017, p. 77–85. URL: http://ieeexplore.ieee.org/document/8099499/. doi:10.1109/CVPR.2017.16.

[9] Qi, CR, Yi, L, Su, H, Guibas, LJ. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. arXiv:170602413 [cs] 2017;URL: http://arxiv.org/abs/1706.02413; arXiv: 1706.02413.

[10] Aumentado-Armstrong, T, Tsogkas, S, Jepson, A, Dickinson, S. Geometric Disentanglement for Generative Latent Shape Models. arXiv:190806386 [cs, eess] 2019;URL: http://arxiv.org/abs/1908.06386; arXiv: 1908.06386.

[11] Cosmo, L, Norelli, A, Halimi, O, Kimmel, R, Rodolà, E. LIMP: Learning Latent Shape Representations with Metric Preservation Priors. In: Vedaldi, A, Bischof, H, Brox, T, Frahm, JM, editors. Computer Vision – ECCV 2020; vol. 12348. Cham: Springer International Publishing. ISBN 978-3-030-58579-2 978-3-030-58580-8; 2020, p. 19–35. URL: https://link.springer.com/10.1007/

978-3-030-58580-8_2. doi:10.1007/978-3-030-58580-8_2; series Title: Lecture Notes in Computer Science.

[12] Crane, K, Weischedel, C, Wardetzky, M. Geodesics in heat: A new approach to computing distance based on heat flow. ACM Transactions on Graphics 2013;32(5):1–11. URL: https://dl.acm.org/doi/10.1145/2516971.2516977. doi:10.1145/2516971.2516977.

[13] Rakotosaona, MJ, Ovsjanikov, M. Intrinsic Point Cloud Interpolation via Dual Latent Space Navigation. In: Vedaldi, A, Bischof, H, Brox, T, Frahm, JM, editors. Computer Vision – ECCV 2020; vol. 12347. Cham: Springer International Publishing. ISBN 978-3-030-58535-8 978-3-030-58536-5; 2020, p. 655–672. URL: https://link.springer.com/10.1007/978-3-030-58536-5_39. doi:10.1007/978-3-030-58536-5_39; series Title: Lecture Notes in Computer Science.

[14] Thomas, H, Qi, CR, Deschaud, JE, Marcotegui, B, Goulette, F, Guibas, L. KPConv: Flexible and Deformable Convolution for Point Clouds. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). Seoul, Korea (South): IEEE. ISBN 978-1-72814-803-8; 2019, p. 6410–6419. URL: https://ieeexplore.ieee.org/document/9010002/. doi:10.1109/ICCV.2019.00651.

[15] Masci, J, Boscaini, D, Bronstein, MM, Vandergheynst, P. Geodesic convolutional neural networks on Riemannian manifolds. arXiv:150106297 [cs] 2015;URL: http://arxiv.org/abs/1501.06297; arXiv: 1501.06297.

[16] Boscaini, D, Masci, J, Rodolà, E, Bronstein, MM. Learning shape correspondence with anisotropic convolutional neural networks. arXiv:160506437 [cs] 2016;URL: http://arxiv.org/abs/1605.06437; arXiv: 1605.06437.

[17] Fey, M, Lenssen, JE, Weichert, F, Müller, H. SplineCNN: Fast Geometric Deep Learning with Continuous B-Spline Kernels. arXiv:171108920 [cs] 2018;URL: http://arxiv.org/abs/1711.08920; arXiv: 1711.08920.

[18] Bouritsas, G, Bokhnyak, S, Ploumpis, S, Zafeiriou, S, Bronstein, M. Neural 3D Morphable Models: Spiral Convolutional Networks for 3D Shape Representation Learning and Generation. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). Seoul, Korea (South): IEEE. ISBN 978-1-72814-803-8; 2019, p. 7212–7221. URL: https://ieeexplore.ieee.org/document/9009455/. doi:10.1109/ICCV.2019.00731.

[19] Ranjan, A, Bolkart, T, Sanyal, S, Black, MJ. Generating 3D Faces Using Convolutional Mesh Autoencoders. In: Ferrari, V, Hebert, M, Sminchisescu, C, Weiss, Y, editors. Computer Vision – ECCV 2018; vol. 11207. Cham: Springer International Publishing. ISBN 978-3-030-01218-2 978-3-030-01219-9; 2018, p. 725–741. URL: http://link.springer.com/10.1007/978-3-030-01219-9_43. doi:10.1007/978-3-030-01219-9_43; series Title: Lecture Notes in Computer Science.

[20] Gong, S, Chen, L, Bronstein, M, Zafeiriou, S. SpiralNet++: A Fast and Highly Efficient Mesh Convolution Operator. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). Seoul, Korea (South): IEEE. ISBN 978-1-72815-023-9; 2019, p. 4141–4148. URL: https://ieeexplore.ieee.org/document/9021965/. doi:10.1109/ICCVW.2019.00509.

[21] Monti, F, Boscaini, D, Masci, J, Rodola, E, Svoboda, J, Bronstein, MM. Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI: IEEE. ISBN 978-1-5386-0457-1; 2017, p. 5425–5434. URL: http://ieeexplore.ieee.org/document/8100059/. doi:10.1109/CVPR.2017.576.

[22] Verma, N, Boyer, E, Verbeek, J. FeaStNet: Feature-Steered Graph Convolutions for 3D Shape Analysis. arXiv:170605206 [cs] 2018;URL: http://arxiv.org/abs/1706.05206; arXiv: 1706.05206.

[23] Zhou, Y, Wu, C, Li, Z, Cao, C, Ye, Y, Saragih, J, et al. Fully Convolutional Mesh Autoencoder using Efficient Spatially Varying Kernels. In: Larochelle, H, Ranzato, M, Hadsell, R, Balcan, MF, Lin, H, editors. Advances in Neural Information Processing Systems; vol. 33. Curran Associates, Inc.; 2020, p. 9251–9262. URL: https://proceedings.neurips.cc/paper/2020/file/68dd09b9ff11f0df5624a690fe0f6729-Paper.pdf.

[24] Bruna, J, Zaremba, W, Szlam, A, LeCun, Y. Spectral Networks and Locally Connected Networks on Graphs. In: 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings. 2014,URL: http://arxiv.org/abs/1312.6203.

[25] Kipf, TN, Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. arXiv:160902907 [cs, stat] 2017;URL: http://arxiv.org/abs/1609.02907; arXiv: 1609.02907.

[26] Vallet, B, Lévy, B. Spectral Geometry Processing with Manifold Harmonics. Computer Graphics Forum 2008;27(2):251–260. URL: https://hal.inria.fr/inria-00331894. doi:10.1111/j.1467-8659.2008.01122.x.

[27] Reuter, M, Wolter, FE, Peinecke, N. Laplace–beltrami spectra as 'shape-DNA' of surfaces and solids. Computer-Aided Design 2006;38(4):342–366. URL: https://doi.org/10.1016/j.cad.2005.10.011. doi:10.1016/j.cad.2005.10.011.

[28] Fiedler, M. Algebraic connectivity of graphs. Czechoslovak Mathematical Journal 1973;23(2):298–305. URL: https://dml.cz/handle/10338.dmlcz/101168. doi:10.21136/CMJ.1973.101168.

[29] Ying, R, You, J, Morris, C, Ren, X, Hamilton, WL, Leskovec, J. Hierarchical Graph Representation Learning with Differentiable Pooling. arXiv:180608804 [cs, stat] 2019;URL: http://arxiv.org/abs/1806.08804; arXiv: 1806.08804.

[30] Bianchi, FM, Grattarola, D, Alippi, C. Mincut pooling in graph neural networks. 2020. URL: https://openreview.net/forum?id=BkxfshNYwB.

[31] Doosti, B, Naha, S, Mirbagheri, M, Crandall, D. HOPE-Net: A Graph-based Model for Hand-Object Pose Estimation. arXiv:200400060 [cs] 2020;URL: http://arxiv.org/abs/2004.00060; arXiv: 2004.00060.

[32] Chen, Z, Kim, TK. Learning Feature Aggregation for Deep 3D Morphable Models. arXiv:210502173 [cs] 2021;URL: http://arxiv.org/abs/2105.02173; arXiv: 2105.02173.

[33] Bogo, F, Romero, J, Pons-Moll, G, Black, MJ. Dynamic FAUST: Registering Human Bodies in Motion. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI: IEEE. ISBN 978-1-5386-0457-1; 2017, p. 5573–5582. URL: http://ieeexplore.ieee.org/document/8100074/. doi:10.1109/CVPR.2017.591.

[34] Mahmood, N, Ghorbani, N, Troje, NF, Pons-Moll, G, Black, M. AMASS: Archive of Motion Capture As Surface Shapes. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). Seoul, Korea (South): IEEE. ISBN 978-1-72814-803-8; 2019, p. 5441–5450. URL: https://ieeexplore.ieee.org/document/9009460/. doi:10.1109/ICCV.2019.00554.