SpecTrHuMS: Spectral transformer for human mesh sequence learning

Clément Lemeunier, Florence Denis, Guillaume Lavoué, Florent Dupont

SpecTrHuMS: Spectral transformer for human mesh sequence learning

Clément Lemeunier, Florence Denis, Guillaume Lavoué, Florent Dupont

SpecTrHuMS: Spectral Transformer for Human Mesh Sequence learning

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

[a]*Univ Lyon, CNRS, INSA Lyon, UCBL, LIRIS, UMR5205, F-69622 Villeurbanne, France*
[b]*Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, UMR5205, F-69622 Villeurbanne, France*
[c]*Univ Lyon, Centrale Lyon, CNRS, INSA Lyon, UCBL, LIRIS, UMR5205, ENISE, F-42023 Saint-Étienne, France*

ARTICLE INFO

Article history:
Received July 6, 2023

Keywords: Geometric Deep Learning, Spectral analysis, Autoencoder, Transformers, Human body triangular mesh sequences

ABSTRACT

We present SpecTrHuMS, a Spectral Transformer for 3D triangular Human Mesh Sequence learning which combines known deep learning models with spectral mesh processing to capture characteristics of 3D shapes as well as temporal dependencies between the frames. Unlike previous works in this field, our approach is able to work directly with a compressed representation of the geometry, the spectral coefficients, rather than relying solely on skeleton joints that does not contain surface information. The vertices of each mesh of a sequence are first projected on the eigenvectors of the Graph Laplacian computed from the common triangulation. A convolutional encoder then encodes each frame into lower dimensional latent variables that preserve as much as possible the spectral information. These latent variables are next passed through a transformer architecture so that the model understands the context of the sequence and learns temporal dependencies between the frames. Each frame of the transformer's output is then decoded by a convolutional decoder which aims to reconstruct the input spectral coefficients. Finally, all frames are transformed back into the spatial domain, resulting in a general process able to treat 4D surfaces with a constant connectivity. Our method is evaluated on a prediction task on AMASS, a dataset of human surface sequences, showing the ability of our model to produce realistic movements while preserving the identity of a subject, and showing that this work is a significant step towards efficient and high-quality representation of triangular mesh sequences with constant connectivity. Additional experiments show that our model can be easily extended to other tasks such as long term prediction, completion and that it is generalizable to other datasets with constant connectivity. This work opens up new possibilities for applications in the fields of animation, virtual reality, and computer graphics. Pretrained models, the code to train them and the code to create datasets are available at https://github.com/MEPP-team/SpecTrHuMS.

1. Introduction

Modeling, analyzing and synthesizing human mesh sequences is a crucial problem in various fields from the computer graphics world such as virtual reality, video games or movies. Recent progress concerning the technology to capture moving shapes made the acquisition cheaper and more efficient while making databases of dynamic meshes more available and more detailed. Nevertheless, the cost and the difficulty to obtain those scans are still non negligible, and developing processes able to understand or generate synthetic but realistic data is today a necessary need.

This last decade, deep learning techniques have been massively used in order to treat large datasets composed of one-

dimensional (e.g. audio, time series and language) or two-dimensional (e.g. images and videos) samples. More recently, a new field has emerged which aims to apply those techniques to data embedded in a higher dimensional space that can be represented as triangular meshes or point clouds and that do not have the Euclidean structure of one or two dimensional data anymore. This field, named Geometric Deep Learning [1], is still the subject of a consequent amount of research, and represents challenges that are the main obstacle when trying to treat sequences of meshes. Recent works concerning human motion modeling deal with this problem by taking as input information regarding sequences of skeletons and not surfaces ([2, 3, 4, 5, 6, 7, 8]), reducing by a large factor the dimension of the problem. The disadvantage of these methods is that they do not have access to the geometry information and need additional skinning and blending steps in order to generate meshes. Other methods [9] take as input a simplified representation of surfaces such as SMPL parameters [10] that contain this geometric information, but they are specific to some datasets and still need an additional process to generate meshes. Also, there are methods able to treat surfaces but they only rely on poses in a static manner, without taking the motion into account [11, 12, 13, 14]. More discussions about the state of the art are given in Section 2. In this work, our goal is to show that by using spectral methods, it is possible to efficiently treat surfaces by also taking into account the dynamic of a motion.

Spectral analysis applications allow to alleviate the amount of treated data by working in the spectral domain. A signal can be well approximated using only low frequencies spectral coefficients, reflecting the energy compaction of the spectral domain. Those properties are good candidates to solve the problems of Geometric Deep Learning, that are the large amount of nodes and their non-ordering.

The process presented here is based on this characteristic of the spectral domain. It consists in a convolutional autoencoder able to represent a static mesh into a latent variable by taking spectral coefficients as input instead of spatial vertices. This architecture is coupled with the recently introduced transformer able to understand the context of sequences from a dataset. It is based on BERT [15] (only the encoder part of the transformer) and is mainly evaluated on the task of predicting the end of a sequence. Additional experiments show the possibility to easily extend its applications, such as long term prediction, completion of missing parts (in-betweening) or generalizability to other datasets. The use of the spectral domain enables a fast treatment of the data, and the transformer enables a parallelizable process that can generate human motion in a non-autoregressive manner (in this context, a non-autoregressive generation means that multiple poses are generated in one pass thanks to the transformer, in contrast with recurrent neural networks that generate outputs based on previous generations, step by step). The main process is depicted in Fig 1.

The contributions of our paper are summarized as follows :

1. we use the spectral coefficients obtained from the geometry of human triangular meshes with constant connectivity as input of the network.
2. we only train the network in the spectral domain without going back to the spatial one, making the process fast.
3. we use a double architecture composed of a convolutional autoencoder and a transformer encoder able to predict future frames (in short or long term), or complete missing frames of a mesh sequence.
4. the latent space of the convolutional autoencoder is used as input and output of the transformer, giving it access to surface information.
5. the process works in a non-autoregressive manner, enabling the efficient generation of human motion frames in a fast way.

Pretrained models, the code to train them and the code to create datasets are available at https://github.com/MEPP-team/SpecTrHuMS.

We present related works in Section 2, we recall spectral mesh processing techniques and introduce how our model works in Section 3, and finally show results of experiments in Section 4.

2. Related works

In this section, we first highlight methods enabling the generation of static triangular meshes which are part of recent studies on Geometric Deep Learning [1]. Then, recent works enabling the treatment of human motion are presented, which allows us to show how our work differs from them by combining spectral triangular mesh processing with temporal modeling.

2.1. Static

Deep learning has proven to be useful when trying to detect features in data that is grid structured like time series or images. Recently, more applications appear where the information is represented as graphs or manifolds such as triangular meshes. Such domains have an increased complexity and bring challenges when trying to transfer known deep learning architectures to them. These challenges are mainly the irregular structure of the grid and the high and variable number of nodes. Here, we focus on methods enabling the generation of human triangular meshes and more specifically movement-like generation by interpolating between two poses. It is difficult to learn on human surfaces in an unsupervised manner with deep learning methods because the human body is subject to near-isometric transformations. The network should be able to understand the manifold of realistic poses of a human body.

A first line of works take as input triangular meshes by treating them as point clouds (refer to [16] for a recent survey). Most of them have an architecture based on PointNet [17, 18]. Aumentado et al. [19] use a variational autoencoder to disentangle intrinsic and extrinsic information using spectral information in a unsupervised way. Cosmo et al. [11] use a strong geometric prior in order to make the latent space preserve computed

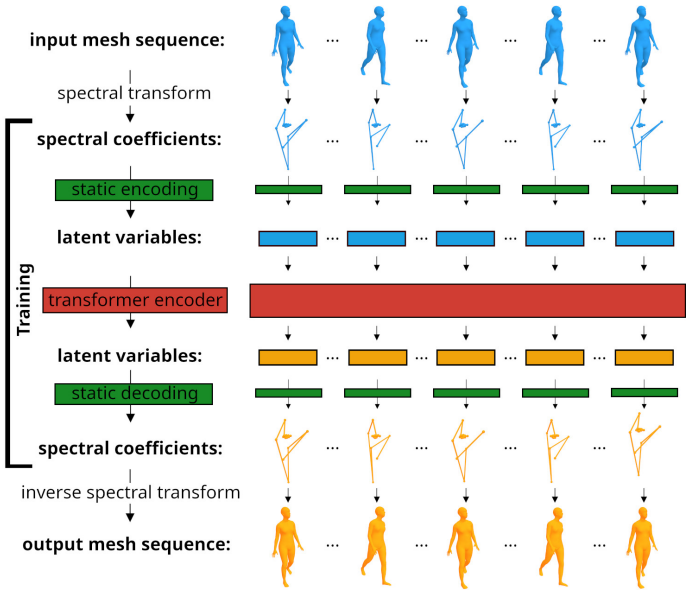

Fig. 1. Illustration of the general process. Each frame's vertices of a mesh sequence are first transformed into spectral coefficients. They are next passed through a convolutional encoder in order to get latent variables, which are the input of a transformer. The latent variables generated by the transformer are then decoded by a convolutional decoder, and finally transformed back to the spatial domain. The training only takes place in the spectral domain, meaning that there is no need to go back to the spatial one during training. Input data are in blue, output data are in orange, learnable parameters for the static process are in green, and learnable parameters for the dynamic process are in red, meaning the overall parameters of the model are in green and red.

geodesic distances on generated surfaces. Rakotosaona et al. [12] use a mapped double latent space, one extracted from vertices and the other one extracted from edge lengths in order to generate movements by interpolating in the edge latent space. Even if these methods result in processes able to treat variable topologies with simple and efficient architectures, they still are slow and only able to treat small datasets having meshes with few vertices. Also, their core architectures based on PointNet are not able to capture local correlation between neighbour vertices and are often less expressive than methods using the connectivity of meshes.

A second line of works takes as input triangular meshes, exploit the information available in the connectivity and take place only in the spatial domain. Pioneer works from Masci et al. [20] and Boscaini et al. [21] used local patches in order to apply convolutions to meshes. Fey et al. [22] pre-defined local pseudo coordinates over the graphs. Lim et al. [23] introduced spiral convolutions, completed by Bouritsas et al. [24] by coupling them with an autoencoder, and finally upgraded by Gong et al. [13]. Hanocka et al. [25] apply convolutions and pooling to meshes by using edges information. Huang et al. [26] introduce a loss based on as-rigid-as-possible (ARAP) deformations which is combined with a network inspired by FeastNet from Verma et al. [27]. Milano et al. [28] aggregate features from edges and faces using attention mechanisms. As for point clouds, the high dimensionality of the data makes processes slow and only able to work on a reduced number of vertices if the dataset contains a lot of meshes, which is the case when treating sequences of meshes. Also, these methods imply to create pooling procedures that are not trivial.

Another line of work use spectral graph theory. By first transforming nodes of a graph in the spectral domain using the eigenvectors of the Graph Laplacian, a convolution layer can be written as a product of signals. One of the first works using this method was introduced by Bruna et al. [29]. But there is a computational disadvantage induced by the forward/inverse Graph Fourier transform, so Defferrard et al. [30] used truncated Chebyshev polynomials and Kipf et al. [31] used only first-order Chebyshev polynomials that resulted in faster processes. This resulted in an autoencoder introduced by Ranjan et al. [32]. Other polynomials were also used, such as Cayley ones by Levie et al. [33] and Zernike ones by Sun et al. [34]. But these methods still go back to the spatial domain for input data processing, slowing the process. Lemeunier et al. [14] are able to encode human triangular meshes by only taking as input the spectral coefficients, without going back to the spatial domain, resulting in a faster process able to give similar results. Marin et al. [35], with its extension [36], presented a model that learns how to reconstruct a 3D shape when only given as input eigenvalues from the Laplace-Beltrami operator, while making it a generative process by using interpolation or style transfer. Similarly, Pegoraro et al. [37] developped a shape from eigenvalues model but while using a mixture of multiple operators. More recently, Sharp et al. [38] use diffusion in the spectral domain to get a model agnostic to sampling, resolution and representation.

Also, deformation-based representations can be exploited to learn on surfaces. Gao et al. [39] designed a representation called rotation-invariant mesh difference (RIMD), which was used as input of Mesh VAE [40] for deformable shape analysis and synthesis. Next, Gao et al. [41] designed an as-consistent-as-possible (ACAP) representation which imposes limitations between adjacent vertices. Then, they combined this representation with graph convolutions [30] in [42] with a newly de-

signed pooling operation.

Finally, other methods were presented such as the one from Eisenberg et al. [43] where a deformation field combined with graph convolutions are used in order to learn interpolation. Also, Hartman et al. [44] use a Riemannian framework for human body scan representation, interpolation and extrapolation.

All these methods have the advantage of achieving generation tasks, but only by considering shapes in a static manner and using methods such as interpolations, reducing the capacity of creation. In this paper, our goal is to combine this way of treating surfaces while including a dynamic treatment. As we work here with mesh sequences, and the resulting datasets consist of a large number of surfaces, we kept the method from Lemeunier et al. [14] for processing the input samples within a reasonable time.

2.2. Dynamic

Human motion generation has recently seen a lot of interest in the litterature. Since our main application is prediction, we will focus on recent papers concerning this objective. Human motion prediction is generally expressed as a sequence to sequence task where the observed motion is represented as the input. First methods used Gaussian process [45], Restricted Boltzmann Machine [46] or Markov models [47]. More recent methods used recurrent neural networks (RNNs) for longer and better predictions [48, 49] but still suffered from discontinuities and were trained only on specific actions. These disadvantages were corrected by working on velocities [50] or with RNNs variants [51, 52]. Still, training or inference is difficult with RNNs and have memory constraints, so prediction was further improved by using sliding windows [53, 54], convolutional models [55, 2] or adverserial training [56, 55]. Next, other works use Graph Convolutional Networks (GCNs [31]) to better encode spatial connectivity of human joints. The first one used DCT coefficients to encode temporal information in the frequency space [3]. Various sizes for convolutional layers coupled with a GCN are used in [57], and variants of GCNs are used in [7, 8, 58]. A fully connected network is coupled with convolutions to encode temporal and spatial information in [53]. Some works use the transformer architecture [59] coupled with a RNN [60], by combining attention to correlate temporal and spatial information [61], or by combining attention with DCT coefficients [62]. Others use attention with GCNs and DCT [4] or by fusing the predictions from three attention modules that process motion at different levels: full body, body parts, and individual joints [5]. Also, motion completion is achieved in [63] by using a transformer architecture. Guo et al. [6] use a simple multilayer perceptron (MLP) and prove that a small and simple model is able to give state of the art results. Another line of work generate human movement conditioned on text by using a variational autoencoder (VAE) [64] or by using diffusion [65]. While conditioning on text is out of the scope of this paper, all cited methods have in common the fact that only skeleton information is exploited, resulting in processes not having access to surface details.

Differently, Marsot et al. [9] use a conditional variational autoencoder (CVAE) [66] in order to create a latent space allowing to represent and generate human motions by taking as input SMPL parameters, thus taking into account surface information. This is similar to our case since these parameters represent a compact space, as the spectral one we use, from which a mesh can be recovered. Nevertheless, frameworks similar to the SMPL one are dedicated to humans, or at least surfaces that can be approximated by skeletons like animals [67]. On the other side, our framework can be easily generalized to other kind of dynamic surfaces that cannot be approximated with skeletons (see Section 4.6.3). Also, the model from Marsot et al. [9] needs a disentangled space as input, while ours have the spectral coefficients space as input, where pose and identity information are entangled.

Finally, Fernandez-Abrevaya et al. [68] introduced a framework that allows to extend the Laplace-Beltrami operator to temporally coherent mesh sequences, thus including surface and temporal information and allowing to edit a mesh sequence in a simple way. While it would be interesting to see how deep learning models could utilize the spectral information coming from this operator, it is impractical to introduce it in our framework: in the same way as using the Laplace-Beltrami operator in a static way, each sequence would necessitate a new computation and eigendecomposition, each time creating a new basis. In our case, by using the Graph Laplacian in a static manner, only one computation of eigenvectors is needed.

In this work, we show that it is possible to develop applications such as human motion prediction while giving as input to the network information on surfaces by using spectral mesh processing coupled with a convolutional autoencoder and a transformer architecture. In the next section, the process of the framework is presented.

3. SpecTrHuMS method

The process we present in this paper is composed of a double network architecture: the first one is the Spectral Autoencoder (SAE) presented in Lemeunier et al. [14], and the second one is a transformer encoder [59]. We first recall notions of spectral mesh processing, how the SAE works, and introduce how we couple it with the second transformer architecture.

3.1. Spectral mesh processing

Surfaces can be studied using spectral mesh processing through operators, usually variants of the Laplacian, in order to obtain a new basis useful for multiple applications. In our case, the properties of the Graph Laplacian are exploited since this operator only rely on the topology of a mesh and thus give the same basis for meshes with the same connectivity, which is the case for the used dataset. If we were to use another operator based on the geometry of each mesh (such as the Laplace-Beltrami), multiple sets of eigenvectors would have been calculated and spectral synchronization would have been necessary.

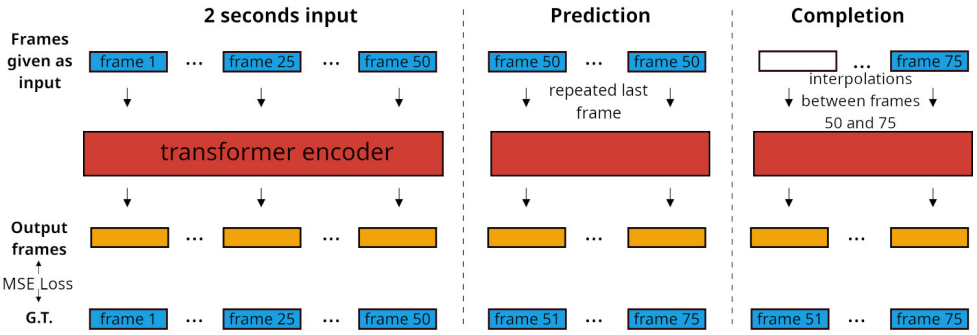

Fig. 2. Process of the transformer. For prediction and completion, 2 seconds, or 50 frames at 25Hz, are given as input to the transformer (on the left). For prediction (in the middle), the last frame of the input sequence is repeated for the ending part of the sequence. For completion (on the right), the ending part is replaced by an interpolation between the last known input frame and the last frame of the sequence. Then, an MSE is computed between the output of the transformer and the ground truth as a loss function.

Each mesh of a dataset with the same connectivity can be projected on the same eigenvectors of the Graph Laplacian in order to obtain their spectral coefficients. To obtain these eigenvectors, the operator L is first computed as $L = D - A$, D being the diagonal matrix of degrees of each vertex and A being the adjacency matrix. The matrix $L \in \mathbb{R}^{n \times n}$, n being the number of vertices, can be decomposed into k pairs of scalar eigenvalues λ_i and eigenvectors ϕ^i with $i \in [1, k]$ and $k <= n$.

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

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

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.

When using enough eigenvectors, this process allows to represent a surface in a compact space, where low frequencies coefficients contain the global information and high frequencies coefficients contain details information. For more details on this process, please refer to the paper [14].

3.2. Spectral autoencoder (SAE)

We use the architecture SAE-LP from Lemeunier et al. [14] that consists in successive layers of convolutions, pooling/upsampling using matrix multiplications and activation layers. The spectral coefficients are the input of this network. This enables the creation of a latent space from which reconstruction is possible.

The static process is as follows: a mesh composed of n vertices is represented as a matrix $V \in \mathbb{R}^{n \times 3}$. The absolute coordinates of the vertices are transformed into spectral coefficients $C \in \mathbb{R}^{k \times 3}$ with a spectral transform: $C = \Phi^T \cdot V$, with $k <= n$ being the number of used eigenvectors. These spectral coefficients C are passed through the SAE's encoder in order to obtain a latent variable $X \in \mathbb{R}^l$, l being the size of the latent space representing a more compact space than the spectral domain. This encoder is made of blocks composed of convolution, downsampling and activation layers, followed by a final fully connected layer. In the next section, we will show how latent variables of a mesh sequence are passed through the transformer architecture. The latent variable of a mesh is then passed through the SAE's decoder in order to obtain the output spectral coefficients $\hat{C} \in \mathbb{R}^{k \times 3}$. The decoder is made of a first fully connected layer followed by blocks composed of upsampling, convolution and activation layers. For visualisation, the reconstructed spectral coefficients are transformed back into the spatial domain to obtain the reconstructed vertices with the matrix multiplication $\hat{V} = \Phi \cdot \hat{C}$ with $\hat{V} \in \mathbb{R}^{n \times 3}$. The choice of the values k and l are precised in section 4.

3.3. Motion transformer

Our goal is to feed a human motion to a neural network so that it understands the context of the movement, which is represented as a sequence of spatial vertices $V^{1:t} \in \mathbb{R}^{t \times n \times 3}$ with t being the number of frames of the sequence. We exploit the widely used transformer architecture for the second part of our network. In order to be able to give the information contained in the sequence of spatial vertices, we first transform each frame in the spectral domain, giving a sequence of spectral coefficients $C^{1:t} \in \mathbb{R}^{t \times k \times 3}$, and then encode each frame of the spectral coefficients sequence with the SAE's encoder introduced before, giving a sequence of latent variables $X^{1:t} \in \mathbb{R}^{t \times l}$.

The generation task is formulated as a sequence-to-sequence problem. For the unknown input latent variables, before feeding them to the transformer model, we either repeat the last frame of the input sequence if the objective is prediction, which is the main task the model is evaluated on, either use linear interpolation to fill in missing values if the objective is completion, which is an additional task. Then, the loss is computed as the error between the output of the transformer and the ground truths in order to train the model. Figure 2 shows how sequences are fed to the model in both cases. Since we subsample each sequence to 25Hz (more information on used datasets are given in Section 4), this means that $t = 75$. For prediction, the latent

variables $X^{1:50}$ are ground truths, and $X^{51:75}$ are X^{50} repeated. For completion, the latent variables $X^{1:50}$ and X^{75} are ground truths, and $X^{51:74}$ are interpolations between X^{50} and X^{75}.

The flow in the transformer part is as follows. First, the input sequence of latent variables $X^{1:t}$ is projected to the transformer's input dimension with a fully connected layer. This gives a set of projected latent variables $X_p^{1:t} \in \mathbb{R}^{t\times d}$, d being the output dimension of the fully connected layer and the input dimension of the transformer.

The design of the transformer architecture makes it unable to know the order of the input frames, so the latent variables are summed with a standard positional encoding $P_e \in \mathbb{R}^{t\times d}$. The positional encoding is a learnable matrix of sinusoidal functions. The resulting embedding of projected latent variables is $X_e^{1:t} = X_p^{1:t} + P_e$, with $X_e^{1:t} \in \mathbb{R}^{t\times d}$.

We use a standard transformer encoder architecture to process sequential embedded latent variables. It is composed of multiple encoder layers, each consisting in multi-head self-attention layers and feed-forward networks, with a residual connection between the two layers and a final norm layer. The multi-head attention layers allow a dense computation between each pair of input frames and long range relations to be captured. We refer the reader to the literature [59] for more details on the general process of this architecture. The transformer encoder outputs a sequence of reconstructed embedded latent variables, which is then projected back to the dimension of the latent space of the static network using fully connected layers, giving a sequence of reconstructed latent variables $\hat{X}^{1:t} \in \mathbb{R}^{t\times l}$. Each frame of those reconstructed latent variables can finally be decoded by the convolutional decoder in order to get reconstructed spectral coefficients, which can then be transformed back to the spatial domain for visualisation.

3.4. Loss functions

The two networks (SAE and the transformer) are trained simultaneously. The first one, represented as a static encoder and decoder, is trained in order to reconstruct the input spectral coefficients without using the dynamic network. The loss to train this static network is thus the mean squared error between input spectral coefficients C and output spectral coefficients $\hat{C}$ without using the transformer. The second dynamic network, the transformer, is trained in order to reconstruct the latent variables sequence. The loss to train this network is thus the mean squared error between the input latent variables sequence $X^{1:t}$ and the output latent variables sequence $\hat{X}^{1:t}$.

4. Experiments

4.1. Baselines

We evaluate against multiple baselines. convSeq2Seq [2] uses a convolutional model, but as for images, convolutions do not consider relations between distant elements, which is important for human behavior understanding. LTD [3] uses GCN for spatial information by building graphs across skeletons joints and DCT coefficients for temporal information, leading to a model already achieving a good prediction. However, such a graph was still insufficient and has been surpassed by Pose Motion Att [4] that uses attention coupled with GCN and DCT, which was upgraded with Motion Att. + Post-fusion [5] by adding the predictions from three other modules operating at full body, body parts, and individual joints levels. It achieved state of the art results until siMLPe [6] introduced a simple multilayer perceptron (MLP) operating on skeletons' joint positions encoded with a Discrete Cosine Transform. STS-GCN [7] and STG-GCN [8] use variants of GCNs, but were also surpassed by siMLPe. First, all these baselines use skeletons' joints information, meaning that their architectures are not directly applicable to surface vertices because of their high dimensionality and their non-ordering. Also, these architectures are only intended for the task of prediction. On the other hand, our architecture is applicable to surfaces even with a high number of vertices since we do not use all available frequencies, and using the transformer architecture makes our process easily generalizable to other tasks such as completion or easily generalizable to training on sequences of different lengths. Finally, using a transformer architecture could be useful, in the future, for more complex learning steps such as pretraining with masked modeling and then fine-tuning with specific losses for other different tasks.

4.2. Datasets

AMASS [69] is a unification of multiple datasets using the SMPL parameterization. We follow [4, 5, 6] and use multiple datasets such as CMU, KIT and other ones (all part of AMASS) as the train set and AMASS-BMLrub as the test set. For the first part of experiments, we follow the state of the art and use a unique identity without hand gestures in order to generate the dataset so that our results are comparable. Models trained on this dataset will have the suffix OI for One Identity. For the second part of experiments, since we work with surfaces and the identity of a mesh has meaning, we generate the same dataset using different identities coming from ground truths in order to show that our network is able to understand the conservation of identity in a human movement. Models trained on this dataset will have the suffix MI for Multiple Identities. We recall that both datasets are composed of the same number of poses, but one is made of an unique identity while the other one is made of multiple identities. Also, since they use the same discretisation from SMPL, the eigenvectors are computed only once and are usable for both of them. Similar to previous works, we ignore the global translation and rotation of the poses and downsample each sequence to 25Hz. After being processed and for visualisation, the sequences are upsampled to 60Hz.

We follow prior works to select the input sequences: we set the input length to 50 frames (2 seconds) and the output length to 25 frames (1 second). These 75 frames are selected using a sliding window over the AMASS sequences with an offset of 5 frames. The train set is composed of 7,799 animations and gives a set of 7,475 animations after filtering since some

Dataset	AMASS-BMLrub							
Time (ms)	80	160	320	400	560	720	880	1000
repeating last frame	24.0	45.0	77.5	89.0	103.7	107.7	101.5	95.3
convSeq2Seq [2]	20.6	36.9	59.7	67.6	79.0	87.0	91.5	93.5
LTD-10-10 [3]	10.3	19.3	36.6	44.6	61.5	75.9	86.2	91.2
LTD-10-25 [3]	11.0	20.7	37.8	45.3	57.2	65.7	71.3	75.2
Pose Motion Att [4]	11.3	20.7	35.7	42.0	51.7	58.6	63.4	67.2
Motion Att. + Post-fusion [5]	11.0	20.3	35.0	41.2	50.7	57.4	61.9	65.8
siMLPe [6]	**10.8**	**19.6**	**34.3**	40.5	50.5	57.3	62.4	65.7
SpecTrHuMS-OI-ES (ours)	15.4	22.3	34.8	**39.9**	**47.6**	**52.8**	**56.9**	**59.6**
SpecTrHuMS-OI (ours)	11.9	21.3	39.1	46.2	56.	63.1	69.4	73.2

Table 1. We compare MPJPE scores for two of our models on the one identity (OI) dataset for prediction at each time step (without taking into account previous predicted frames), one with early stopping (ES) and the second one without early stopping. When early stopping, our model is able to give better results than the state of the art for long time prediction.

Dataset	AMASS-BMLrub							
Time (ms)	80	160	320	400	560	720	880	1000
repeating last frame	18.3	29.1	47.7	55.4	67.9	76.6	81.7	83.5
STS-GCN [7]	10.0	12.5	21.8	24.5	31.9	38.1	42.7	45.5
STG-GCN [8]	10.0	11.9	20.1	24.0	30.4	-	-	43.1
siMLPe [6]	**6.1**	**10.8**	**19.1**	**22.8**	**29.5**	**35.1**	**39.7**	42.7
SpecTrHuMS-OI-ES (ours)	13.9	17.2	23.8	26.8	32.	36.2	**39.7**	**42.**
SpecTrHuMS-OI (ours)	9.8	14.3	23.5	27.7	34.8	40.5	45.4	48.6

Table 2. We compare MPJPE scores for two of our models on the one identity (OI) dataset for average prediction at each time step (by taking into account previous predicted frames), one with early stopping (ES) and the second one without early stopping. When early stopping, our model is able to give better results than the state of the art for long time prediction.

animations are shorter than the needed window, giving 458,104 windows, which is equivalent of approximately 380 hours of video. The test set is composed of 3,061 animations and gives 2,640 animations after filtering and 145,443 windows.

4.3. Evaluation metrics

As we focus on surfaces while previous works only examine skeletons, a direct comparison is not feasible. To align with the current state of the art, we begin by presenting the results using the Mean Per Joint Position Error (MPJPE) [70] calculated on 3D skeleton joint coordinates. We obtain these joints positions by using the joint regressor available in the SMPL parameterization that allows to transform mesh vertices to 3D joints positions. But this MPJPE only compares generated movements with a ground truth, and a generated motion could still be realistic without having a good score following this metric. So, since the MPJPE does not perfectly reflect the behaviour of generating realistic movements, we also introduce an edge length variation measure that expresses the conservation of the identity. This measure is computed as the absolute mean pairwise difference between corresponding edge lengths of the last known frame of the input motion with predicted frames so that it does not depend on the ground truth but rather on the conservation of identity during the movement. This measure will be named Mean Edge Length Variation (MELV).

4.4. Implementation

The number of mesh vertices coming from SMPL is $n = 6890$. We set the value of used eigenvectors to $k = 512$ (on the 6890 available) since it is sufficient to have enough detailed surfaces and since it alleviates the work of the coupled network. The static network is composed of 3 layers for the encoder and decoder each, and uses a convolution window of size 3. The latent size of the static network is set to 256, which is the same as the input dimension of the transformer. The transformer is composed of 4 heads, 6 layers, has a feedforward dimension of 1024 and has the GELU function as activation.

The spectral coefficients are first precomputed and stored locally on the hard-drive. We standardize (subtraction of mean and division by the standard deviation) the input of the static network but compute the loss on the destandardize spectral coefficients so that more importance is given to low frequencies (we recall that low frequencies spectral coefficients have higher magnitudes than high frequencies ones, see [14]). The input latent variables are also standardized and the loss of the transformer is computed on these standardized latent variables. The static and dynamic networks are trained simultaneously. The static network is first trained alone for two epochs so that it first learns to reconstruct approximately the static meshes. At the third epoch, the training of the dynamic network starts, and both networks continue to train. Because the weights of the static network are updated during training, the values of the latent variables are modified, so we use the Welford algorithm [71] to update the means and standard deviations of the latent variables during training.

We use the PyTorch framework to train the networks for 100 epochs, with the ADAM optimizer, with a batch size of 32 and a starting learning rate of 1e-4 without warm-up but with a scheduler that reduces the learning rate to 1e-6 at the end of training. Training is done on a NVIDIA V-100 GPU. More details about the number of epochs are given in the quantitative results section. Training takes approximately 12 hours.

We first present results on the main task of prediction using the dataset that follows previous compared works with one identity and using the dataset with multiple identities. Then, we present other applications easily implemented from our framework such as long term prediction (by using an autoregressive generation), completion (also called in-betweening) and we finally show that our model can be used with other datasets made of surfaces that cannot be approximated by skeletons.

4.5. Main application: prediction

The main task presented in this work is prediction: the model tries to generate a 1 second motion when having as input the previous 2 seconds (see Figure 2 in the middle). In order to be comparable with the literature, we first evaluate our method on a dataset with only one neutral identity. With this dataset, we present two models: one that has been trained for a few epochs (SpecTrHuMS-OI-ES for SpecTrHuMS using One Identity with Early Stopping), and another one that has been

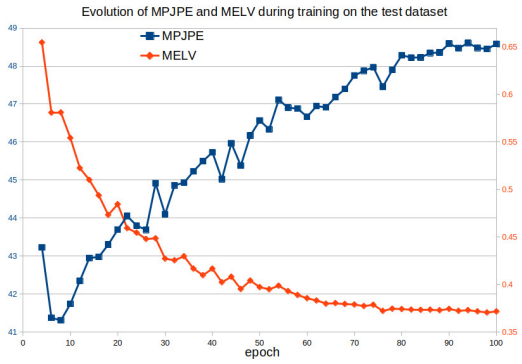

Fig. 3. Curves of the evolution of the mean MPJPE and the mean MELV over all time steps during training on the test set. We can see that the model becomes gradually less accurate in terms of MPJPE but becomes better at preserving edge lengths.

fully trained (SpecTrHuMS-OI). Next, results on the dataset using multiple identities are highlighted with a model named SpecTrHuMS-MI for SpecTrHuMS using Multiple Identities.

4.5.1. Quantitative results

Table 1 showcases the results of our models compared to those of prior studies, employing the evaluation methodology proposed in [4, 5, 6]. In this table, the evaluation score only accounts for predicted time steps and does not take into account previous time steps predictions. Table 2 presents the results obtained through the protocol employed in [7, 8], where evaluations are computed by taking the average over previous frames. Values for other models are taken from prior papers. We also report in Tables 1 and 2 results of MPJPE when simply repeating the last known frame as a baseline. As a reminder, our focus is on surfaces, whereas the methods being compared operate on positions of skeleton joints. Consequently, our network processes additional information that incorporates the identity particulars of the surfaces. While our results are less accurate for shorter time steps, our network, when early stopped, is able to better predict the movement at longer time steps using the MPJPE metric. However, we are going to see that after being fully trained, the model fails to produce better results using this metric, indicating that it learned to retain other features of the data during the training process.

Figure 3 illustrates the progression of evaluation metrics. Notably, the MPJPE begins to increase at epoch 5, while the MELV continues to decrease. This is attributed to the network generating movements that do not precisely replicate the ground truth, but still maintain better consistency in the edge length of the subject. This phenomenon highlights the ambiguity in evaluating the generation of movement, which goes beyond a simple comparison of the predicted and actual positions of skeleton joints. Table 3 presents the MELV values and their corresponding standard deviations for the same model at the epoch yielding the best MPJPE results and the last epoch compared to ground truth values. The results demonstrate that with longer training, the fully trained model is capable of preserving edge lengths more effectively than the early stopped model.

Fig. 4. Visual comparison of our model with one identity early stopped (SpecTrHuMS-OI-ES) with our model fully trained (SpecTrHuMS-OI) on the test dataset. While the early stopped model gives better MPJPE scores, the fully trained one is able to better preserve edge lengths. Blue meshes represent the ground truths with an MELV of 0.72mm. Orange meshes represent the output of our models, with an MELV of 2.03mm for the SpecTrHuMS-OI-ES and 0.59mm for the SpecTrHuMS-OI. All MELVs indicated in this caption are for the last time step.

We now illustrate how our model correctly learns to preserve the identity of a mesh while making it move. We use for this a similar dataset than the previous one but by introducing the variability of the identity (with the suffix MI for Multiple Identities). To the best of our knowledge, there is no existing research that focuses on a prediction task and that directly generates surfaces, making it impossible to compare the quality of our generated surfaces with those of prior works. However, we present in Table 3 that our model, when utilizing multiple identities, achieves similar results in preserving edge lengths when compared to the models that employ only one identity. The conservation of edge lengths in both datasets is approximately the same, as indicated by the ground truth lines, signifying that the values are comparable. More precisely, we can observe that the MELV's evolution slows down significantly at 560ms at approximately 0.5mm, indicating that the evaluation should converge towards this value. We note that the average edge length of the dataset with multiple identities is around 16mm.

4.5.2. Qualitative results

In Figures 4 and 5, we compare visual results of predictions on the test set with one identity when using a model stopped early and giving best results in terms of MPJPE and a model fully trained. MELVs are indicated in Figure's captions for the ground truths and model's predictions. By observing these values and visual representations in both Figures, it is apparent that

Time (ms)	80	160	320	400	560	720	880	1000
Dataset	AMASS-BMLrub-OI (one identity)							
ground truth	0.13 ± 0.08	0.23 ± 0.13	0.38 ± 0.2	0.43 ± 0.22	0.49 ± 0.25	0.51 ± 0.26	0.49 ± 0.28	0.47 ± 0.28
SpecTrHuMS-OI-ES	0.4 ± 0.17	0.48 ± 0.21	0.63 ± 0.31	0.7 ± 0.35	0.81 ± 0.43	0.87 ± 0.47	0.89 ± 0.5	0.88 ± 0.52
SpecTrHuMS-OI	0.2 ± 0.09	0.29 ± 0.13	0.42 ± 0.2	0.47 ± 0.21	0.54 ± 0.23	0.57 ± 0.24	0.57 ± 0.26	0.57 ± 0.27
Dataset	AMASS-BMLrub-MI (multiple identities)							
ground truth	0.13 ± 0.08	0.24 ± 0.14	0.39 ± 0.21	0.44 ± 0.23	0.5 ± 0.26	0.52 ± 0.27	0.5 ± 0.29	0.48 ± 0.29
SpecTrHuMS-MI	0.37 ± 0.14	0.43 ± 0.15	0.55 ± 0.2	0.59 ± 0.21	0.65 ± 0.23	0.68 ± 0.24	0.68 ± 0.26	0.67 ± 0.27

Table 3. Top: comparison of the MELV on the dataset with one identity between the ground truth, a model early stopped (SpecTrHuMS-OI-ES) and a model fully trained (SpecTrHuMS-OI). The fully trained model is able to better preserve edge lengths than the early stopped one. Bottom: comparison of the MELV on the dataset with multiple identities between the ground truth and a fully trained model (SpecTrHuMS-MI). Edge length conservation between the two ground truths for both dataset approximately have the same values, so evaluation on both datasets are comparable. The model trained on multiple identities is able to correctly preserve edge length when compared with SpecTrHuMS-OI.

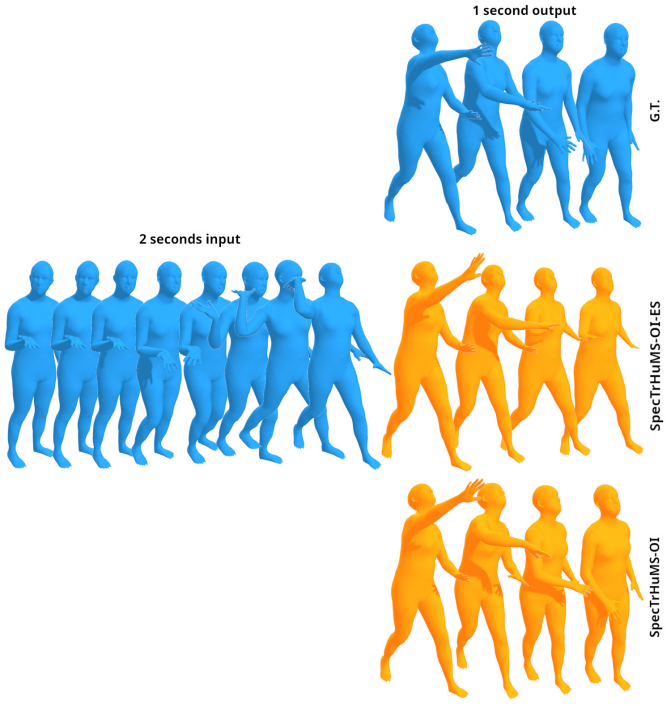

Fig. 5. Visual comparison of our model with one identity early stopped (SpecTrHuMS-OI-ES) with our model fully trained (SpecTrHuMS-OI) on the test dataset. While the early stopped model gives better MPJPE scores, the fully trained one is able to better preserve edge lengths. Blue meshes represent the ground truths with an MELV of 0.96mm. Orange meshes represent the output of our models, with an MELV of 2.31mm for the SpecTrHuMS-OI-ES and 0.64mm for the SpecTrHuMS-OI. All MELVs indicated in this caption are for the last time step.

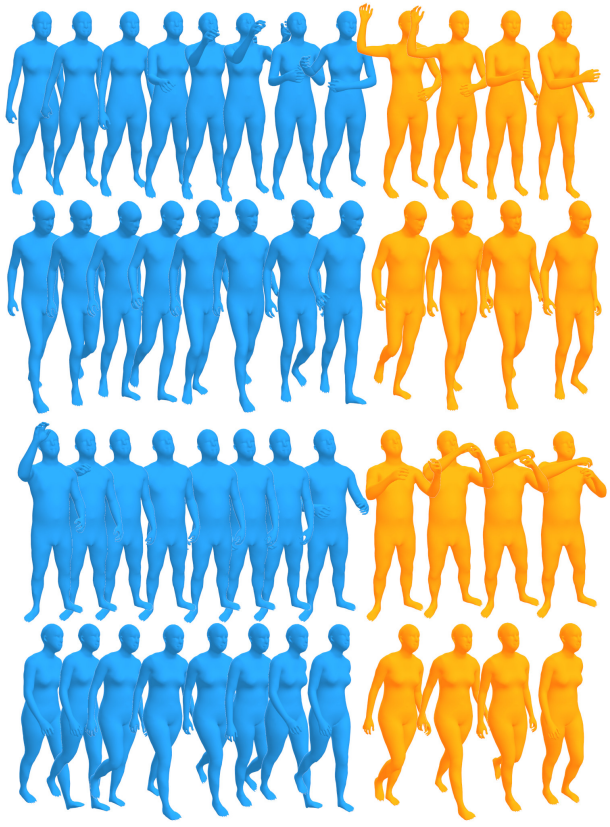

Fig. 6. Visual results with a model trained on a dataset with multiple identities (SpecTrHuMS-MI): the identity of shapes is well preserved while generating a realistic movement.

the model trained for a longer duration exhibits superior ability in maintaining the length of the arms.

In Figure 6, we show visual results with the model using the dataset with multiple identities (SpecTrHuMS-MI). Both Figures show the ability of our model to correctly preserve the identity. Similarly to the model trained on the dataset with one identity, the generated movement is not necessarily close to the ground truth but is still realistic. Nevertheless, the model is able to understand how to move spectral coefficients, and thus the vertices of input meshes, while maintaining the appearance of the subject, which is a crucial feature for many real-world applications where identity preservation is essential.

To summarize, the model exhibiting the best results is the one fully trained even if its score regarding the MPJPE is worse. This is visually evidenced in Figures 4 and 5: when arms lengths are not conserved, the MPJPE is better, meaning that this metric is not suitable for our approach. This is why the MELV metric is more relevant (see Figure 3). This results in predictions that can deviate from ground truths but that are more realistic. In the future, metrics regarding the realism of a movement should be introduced.

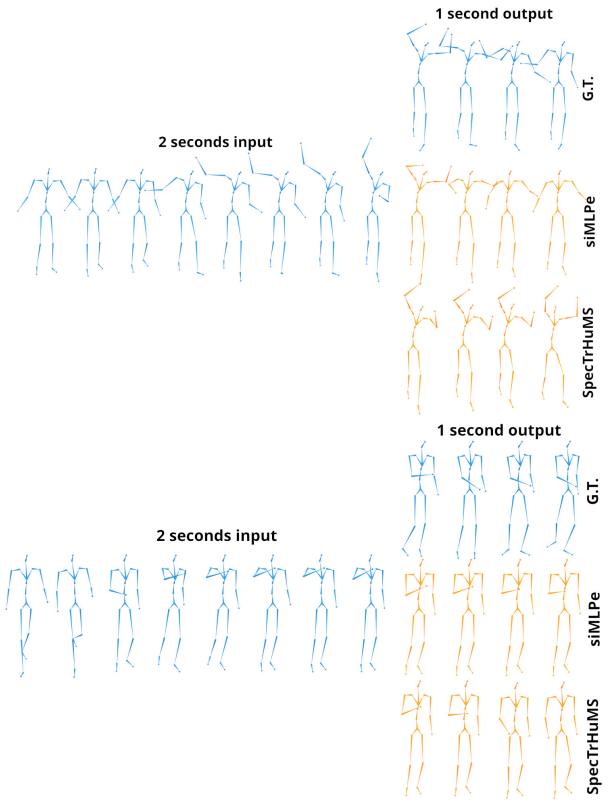

Fig. 7. Visual comparisons with siMLPe [6]. In our case, we get joints positions from mesh vertices using the regression matrix available in the SMPL framework, whereas the baseline directly works with skeletons.

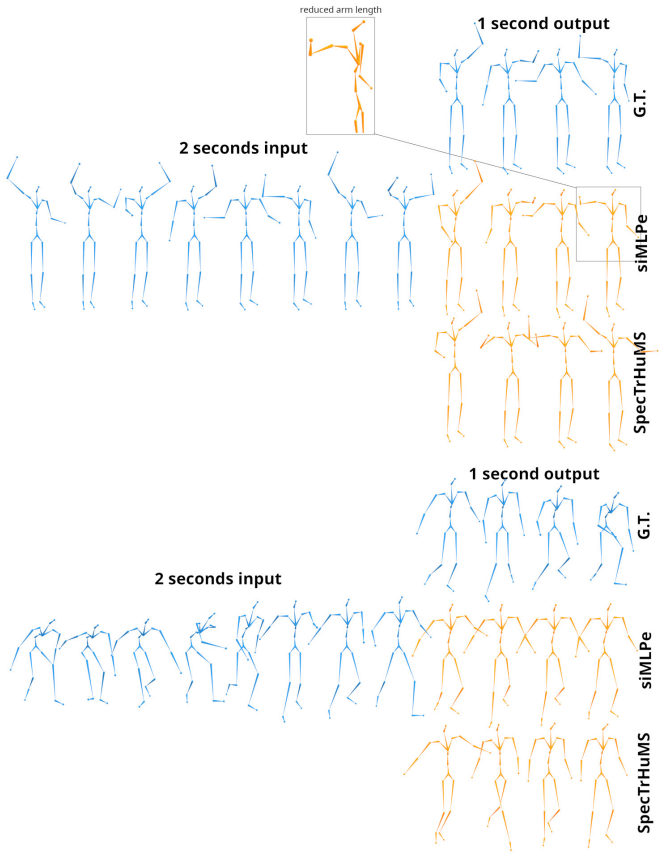

Fig. 8. Visual comparisons with siMLPe [6]. In our case, we get joints positions from mesh vertices using the regression matrix available in the SMPL framework, whereas the baseline directly works with skeletons.

We also present visual comparisons with siMLPe [6] in Figures 7 and 8. We recall that their architecture takes as input skeleton joints whereas our model takes as input spectral coefficients containing surface information (we get joints positions from mesh vertices using the regression matrix available in the SMPL framework). In Figure 7, for the first example, the motion predicted by siMLPe is closer to the ground truth, while the movement predicted by our model deviates from it while still being realistic, reflecting the ambiguity of the MPJPE evaluation approached previously. In other examples, both works tend to produce less dynamic movements than ground truths, but our method is able to generate more movements than siMLPe (see Figures 7 and 8 at the bottom). Also, in the case of siMLPe, the generated skeletons can have reduced arms lengths (see Figure 8 line 2).

An additional video file is provided in order to better visualise results.

4.6. Other applications

We show in this section additional results for other applications, first on a task of long term prediction that does not need another training method, on a task of completion/in-betweening that needs another training method, and finally on a prediction task when using a dataset made of surfaces that can not be approximated by skeletons.

4.6.1. Long term prediction

In order to predict for a longer term a given movement, we adopt an autoregressive method. This approach involves feeding an initial set of 50 frames of motion into the model and predicting the next 25 frames based on the input. The predicted frames are then appended to the initial set, and the process is repeated with the last second of the input motion and the newly predicted frames serving as the new input. By iteratively repeating this process, we are able to generate a sequence of frames that extrapolates the original movement over an extended period of time. Figure 9 shows examples of this application. The first line show long term prediction of arms movement. On the second line, a walking animation is generated, similar to the beginning of the input. The model is able to correctly generate a motion while keeping the movement and identity information. It is important to note that the generation of additional frames with this autoregressive method can be realised in real-time.

4.6.2. Completion / in-betweening

Because our approach utilizes a transformer architecture, it can be easily applied to other tasks such as completion. In this additional application, which needs another training, the model endeavors to predict the movement between two known points, with the provision of two seconds of motion and an additional last frame. Adapting our approach to in-betweening only requires the addition of an extra frame to the input and replacing

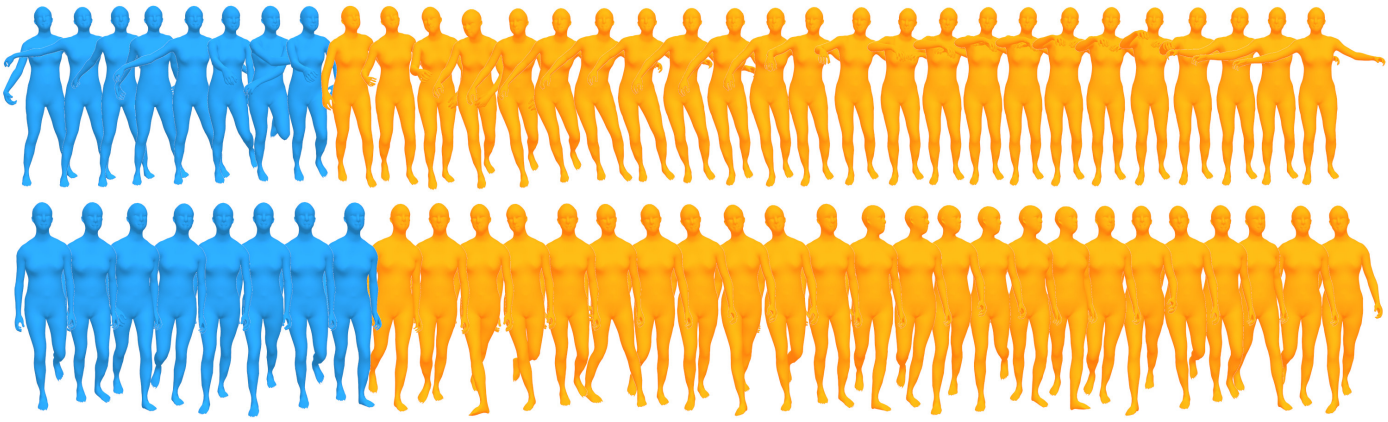

Fig. 9. Examples of extrapolation. Blue meshes are given as input to the model, and orange meshes are generated in an autoregressive manner: 2 seconds are given as input, 1 second is generated, and the next seconds are generated by giving as input the previous generated seconds. The used model is SHTMS-MI, and a total of 6 seconds are generated. The model is able to extrapolate an input motion while keeping the identity and the dynamic information.

Fig. 10. Examples of completion/in-betweening. The model is able to correctly interpolate between two frames while taking into account previous motion.

unknown latent variables with an interpolation between the last frame of the input and the extra frame (see Figure 2). Figure 10 shows example of this application. The model is able to correctly interpolate between two frames while taking into account previous motion and while preserving edge lengths.

4.6.3. Application to other datasets

In order to prove the generalizability of our method, we present experiments on the prediction of a piece of cloth's simulation. Using Blender [72], we generate a triangular mesh representing a piece of cloth (see Figure 11) made of 484 vertices, while assigning it a cloth modifier. Using this mesh, 20,000 simulations of two seconds are created in which two different random vertices are pinned and the surface undergoes the gravity and self-collisions (16,000 are used for training and 4,000 for testing). The mesh is geometrically asymmetrical so that generated simulations are always different. The Graph Laplacian with its eigenvectors are then computed from the connectivity, allowing to create sequences of spectral coefficients that are given as input to our model, in the same way as in Figures 1 and 2 in the case of prediction, but with one second of known input and one second of prediction.

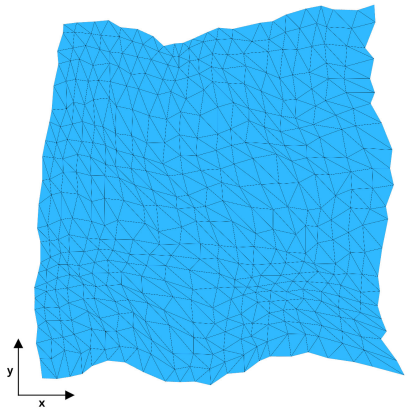

Fig. 11. The used triangular mesh to create a dataset of piece of cloth simulations. It is flat (only on x and y axis) and made of 484 vertices. For each animation, two random vertices are pinned and the fabric undergoes gravity and self-collisions. The mesh is geometrically asymmetrical so that simulations are always different.

Time (ms)	80	160	320	400	560	720	880	1000
Dataset	AMASS-BMLrub-OI							
SpecTrHuMS-OI	0.00590	0.01123	0.02166	0.02575	0.03111	0.03476	0.03782	0.03959
Dataset	Cloth dataset							
SpecTrHuMS-cloth	0.00625	0.00672	0.00769	0.00819	0.00935	0.01069	0.01228	0.01369

Table 4. Comparison of RMSEs on mesh vertices for models trained on the human dataset with One Identity (AMASS-BMLrub-OI) and on the cloth dataset. RMSEs are normalised by the largest bounding box of each mesh so that values are comparable. Values are lower for the cloth dataset, showing that our model is generalizable to cloth simulations.

In Table 4, quantitative evaluations present the root-mean-square error (RMSE) between ground truths and generated surfaces at each time step on the test dataset. RMSE is used since errors are normalised by the largest diagonal length of the bounding box of each mesh so that they are comparable, meaning that the scores are expressed as a percentage of the corresponding mesh's bounding box. The values corresponding to the cloth dataset are comparable with those from the human dataset, showing that our model is generalizable to dynamic surfaces that cannot be approximated with skeletons and that our model does not have to rely on parameterizations such as

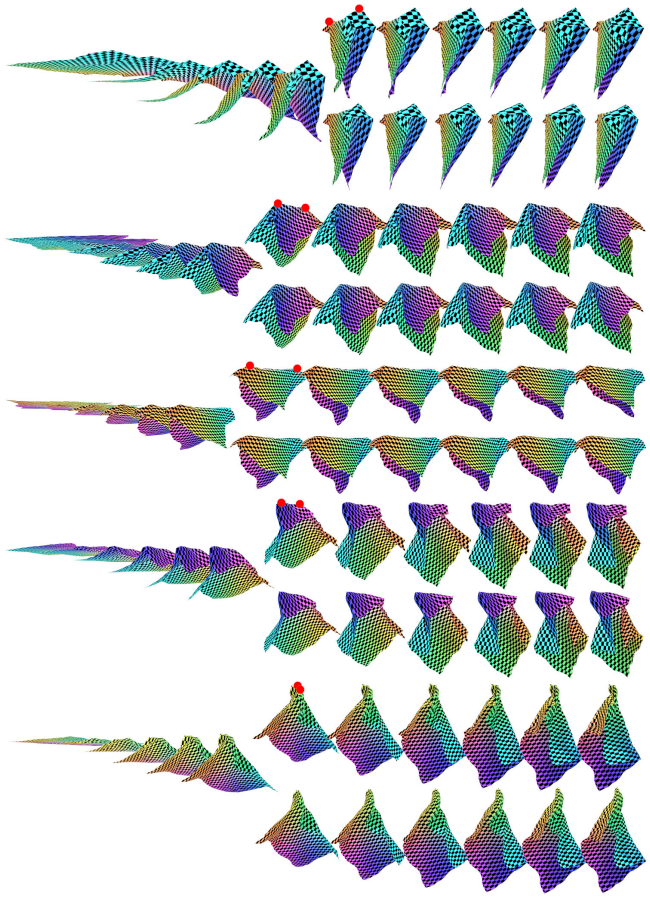

Fig. 12. Visual examples of cloth simulations predictions. For each line, one second is given as input (on the left), on the top are ground truths, and at bottom are generated meshes. For each animation, two random vertices are pinned (marked with red dots) and the cloth undergoes gravity and self-collisions. Our model is able to reproduce realistic movements, self-collisions and folds.

SMPL.

Also visual examples are presented in Figure 12. We can see that the model manages to predict realistic movement of a cloth simulation given the beginning of a sequence. It is difficult to see as the examples are shown as images, but self-collisions are well reproduced in generated animations. Also, created folds are not exactly like in ground truths, showing that the model did not completely learn the dataset but is rather able to generate realistic behaviour of the cloth under gravity. An additional video file is provided in order to better visualise results.

5. Discussion

We showed that our model is able to generate human motion depending on past frames in short and long term, to complete a missing part in a motion, and that it is generalizable to other kind of surfaces without having to rely on parameterizations. Nevertheless, the proposed architecture exhibits some limitations. First, since we use the Graph Laplacian, only datasets made of meshes with a constant connectivity can be given as input. In the case of a dataset with a varying number of vertices and varying connectivity, we could preprocess and remesh all samples with a common connectivity so that it is compatible with our framework. Also, it could be possible to compute the operator for all connectivities and map them with Functional Maps [73] in order to align all different bases to a common one. Secondly, we only used 512 frequencies on the 6890 available from the SMPL discretisation. This leads to meshes that are low-pass filtered and do not show high frequencies details, especially on the face, hands and feet. This is a limitation, but also an advantage since when analysing the motion of a human body, high frequency details are not essential, and our method allows a control over the precision the network has access to. Giving high frequencies to our model is straightforward, while methods such as SMPL need an upgrade to increase the quality of meshes [74]. In the future, this aspect could be improved either by working with more frequencies or by introducing an additional neural network whose task is to complete high frequency details depending on the available low frequency information.

6. Conclusion

In this paper, we presented SpecTrHuMS, a Spectral Transformer for 3D triangular Human Mesh Sequence learning that can efficiently and accurately process 3D triangular mesh sequences. Our model is able to capture both the spatial and temporal dependencies of the shapes, and it does so by directly working with a compressed representation of the spectral information of the shapes. Unlike most of previous works in this field, our model does not rely solely on skeleton joints and is able to preserve surface information of the shapes. Also, compared to works that take as input SMPL parameters which contain surface information, our model is able to have a control over the amount of information it has access to and is generalizable to other datasets that cannot be represented using skeletons.

We evaluated our model on a main prediction task on AMASS, a dataset of human surface sequences. Also, due to the used architecture, we were able to show additional applications that can be easily implemented from our framework. Our experiments show that our model is able to correctly generate a sequence by preserving the meshes' edge lengths, producing realistic movements while preserving the identity of a subject. These results demonstrate the effectiveness of our approach in capturing both physical attributes of an object and features variations over time of 3D triangular mesh sequences. This suggests that our model represents a significant step forward in the efficient and high-quality representation of triangular mesh sequences, which opens up new possibilities for applications in the fields of animation, virtual reality, and computer graphics. Overall, our approach of combining spectral information with a convolutional autoencoder and a transformer provides a promising direction for future works in the field of 4D shape analysis and processing. We believe that our model has the potential to contribute to a wide range of applications and will serve as baselines for future works on this subject. In the future, efforts

will be directed towards training on more extensive databases and adapting the framework to enable users to influence the generation process. This will include modifying the number of spectral coefficients utilized to generate surfaces with varying degrees of detail, as well as allowing users to input specific text to generate targeted actions, rather than relying solely on movement inputs.

Acknowledgments

This work was supported by the ANR project Human4D ANR-19-CE23-0020 and was granted access to the AI resources of IDRIS under the allocation 2023-AD011012424R2 made by GENCI.

References

[1] 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.

[2] Li, C, Zhang, Z, Lee, WS, Lee, GH. Convolutional sequence to sequence model for human dynamics. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE; 2018,URL: https://doi.org/10.1109/cvpr.2018.00548. doi:10.1109/cvpr.2018.00548.

[3] Mao, W, Liu, M, Salzmann, M, Li, H. Learning trajectory dependencies for human motion prediction. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE; 2019,URL: https://doi.org/10.1109/iccv.2019.00958. doi:10.1109/iccv.2019.00958.

[4] Mao, W, Liu, M, Salzmann, M. History repeats itself: Human motion prediction via motion attention. In: Computer Vision – ECCV 2020. Springer International Publishing; 2020, p. 474–489. URL: https://doi.org/10.1007/978-3-030-58568-6_28. doi:10.1007/978-3-030-58568-6_28.

[5] Mao, W, Liu, M, Salzmann, M, Li, H. Multi-level motion attention for human motion prediction. International Journal of Computer Vision 2021;129(9):2513–2535.

[6] Guo, W, Du, Y, Shen, X, Lepetit, V, Alameda-Pineda, X, Moreno-Noguer, F. Back to MLP: A simple baseline for human motion prediction. In: 2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). IEEE; 2023,URL: https://doi.org/10.1109/wacv56688.2023.00479. doi:10.1109/wacv56688.2023.00479.

[7] Sofianos, T, Sampieri, A, Franco, L, Galasso, F. Space-time-separable graph convolutional network for pose forecasting. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE; 2021,URL: https://doi.org/10.1109/iccv48922.2021.01102. doi:10.1109/iccv48922.2021.01102.

[8] Zhong, C, Hu, L, Zhang, Z, Ye, Y, Xia, S. Spatio-temporal gating-adjacency GCN for human motion prediction. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2022,URL: https://doi.org/10.1109/cvpr52688.2022.00634. doi:10.1109/cvpr52688.2022.00634.

[9] Marsot, M, Wuhrer, S, Franco, JS, Durocher, S. A structured latent space for human body motion generation 2021;arXiv:2106.04387.

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

[11] Cosmo, L, Norelli, A, Halimi, O, Kimmel, R, Rodolà, E. LIMP: Learning latent shape representations with metric preservation priors. In: Computer Vision – ECCV 2020. Springer International Publishing; 2020, p. 19–35. URL: https://doi.org/10.1007/978-3-030-58580-8_2. doi:10.1007/978-3-030-58580-8_2.

[12] Rakotosaona, MJ, Ovsjanikov, M. Intrinsic point cloud interpolation via dual latent space navigation. In: Computer Vision – ECCV 2020. Springer International Publishing; 2020, p. 655–672. URL: https://doi.org/10.1007/978-3-030-58536-5_39. doi:10.1007/978-3-030-58536-5_39.

[13] 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). IEEE; 2019,URL: https://doi.org/10.1109/iccvw.2019.00509. doi:10.1109/iccvw.2019.00509.

[14] Lemeunier, C, Denis, F, Lavoué, G, Dupont, F. Representation learning of 3d meshes using an autoencoder in the spectral domain. Computers &amp Graphics 2022;107:131–143. URL: https://doi.org/10.1016/j.cag.2022.07.011. doi:10.1016/j.cag.2022.07.011.

[15] Devlin, J, Chang, MW, Lee, K, Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding 2018;arXiv:1810.04805.

[16] Guo, Y, Wang, H, Hu, Q, Liu, H, Liu, L, Bennamoun, M. Deep learning for 3d point clouds: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 2021;43(12):4338–4364. URL: https://doi.org/10.1109/tpami.2020.3005434. doi:10.1109/tpami.2020.3005434.

[17] Qi, CR, Su, H, Mo, K, Guibas, LJ. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017,.

[18] Qi, CR, Yi, L, Su, H, Guibas, LJ. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17; Red Hook, NY, USA: Curran Associates Inc. ISBN 9781510860964; 2017, p. 5105–5114.

[19] Aumentado-Armstrong, T, Tsogkas, S, Jepson, A, Dickinson, S. Geometric disentanglement for generative latent shape models. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE; 2019,URL: https://doi.org/10.1109/iccv.2019.00827. doi:10.1109/iccv.2019.00827.

[20] Masci, J, Boscaini, D, Bronstein, MM, Vandergheynst, P. Geodesic convolutional neural networks on riemannian manifolds. In: 2015 IEEE International Conference on Computer Vision Workshop (ICCVW). IEEE; 2015,URL: https://doi.org/10.1109/iccvw.2015.112. doi:10.1109/iccvw.2015.112.

[21] Boscaini, D, Masci, J, Rodoià, E, Bronstein, M. Learning shape correspondence with anisotropic convolutional neural networks. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. NIPS'16; Red Hook, NY, USA: Curran Associates Inc. ISBN 9781510838819; 2016, p. 3197–3205.

[22] Fey, M, Lenssen, JE, Weichert, F, Muller, H. SplineCNN: Fast geometric deep learning with continuous b-spline kernels. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE; 2018,URL: https://doi.org/10.1109/cvpr.2018.00097. doi:10.1109/cvpr.2018.00097.

[23] Lim, I, Dielen, A, Campen, M, Kobbelt, L. A simple approach to intrinsic correspondence learning on unstructured 3d meshes. In: Lecture Notes in Computer Science. Springer International Publishing; 2019, p. 349–362. URL: https://doi.org/10.1007/978-3-030-11015-4_26. doi:10.1007/978-3-030-11015-4_26.

[24] 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). IEEE; 2019,URL: https://doi.org/10.1109/iccv.2019.00731. doi:10.1109/iccv.2019.00731.

[25] Hanocka, R, Hertz, A, Fish, N, Giryes, R, Fleishman, S, Cohen-Or, D. MeshCNN. ACM Transactions on Graphics 2019;38(4):1–12. URL: https://doi.org/10.1145/3306346.3322959. doi:10.1145/3306346.3322959.

[26] Huang, Q, Huang, X, Sun, B, Zhang, Z, Jiang, J, Bajaj, C. Arapreg: An as-rigid-as possible regularization loss for learning deformable shape generators. 2021. arXiv:arXiv:2108.09432.

[27] Verma, N, Boyer, E, Verbeek, J. FeaStNet: Feature-Steered Graph Convolutions for 3D Shape Analysis. In: CVPR - IEEE Conference on Computer Vision & Pattern Recognition. Salt Lake City, United States; 2018,URL: https://hal.inria.fr/hal-01540389.

[28] Milano, F, Loquercio, A, Rosinol, A, Scaramuzza, D, Carlone, L. Primal-dual mesh convolutional neural networks. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS'20; Red Hook, NY, USA: Curran Associates Inc. ISBN 9781713829546; 2020,.

[29] Bruna, J, Zaremba, W, Szlam, A, Lecun, Y. Spectral networks and locally connected networks on graphs. In: International Conference on Learning Representations (ICLR2014), CBLS, April 2014. 2014,.

[30] Defferrard, M, Bresson, X, Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. NIPS'16; Red Hook, NY, USA: Curran Associates Inc. ISBN 9781510838819; 2016, p. 3844–3852.

[31] Kipf, TN, Welling, M. Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations. 2017,URL: https://openreview.net/forum?id=SJU4ayYgl.

[32] 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. Cham: Springer International Publishing. ISBN 978-3-030-01219-9; 2018, p. 725–741.

[33] Levie, R, Monti, F, Bresson, X, Bronstein, MM. CayleyNets: Graph convolutional neural networks with complex rational spectral filters. IEEE Transactions on Signal Processing 2019;67(1):97–109. URL: https://doi.org/10.1109/tsp.2018.2879624. doi:10.1109/tsp.2018.2879624.

[34] Sun, Z, Rooke, E, Charton, J, He, Y, Lu, J, Baek, S. ZerNet: Convolutional neural networks on arbitrary surfaces via zernike local tangent space estimation. Computer Graphics Forum 2020;39(6):204–216. URL: https://doi.org/10.1111/cgf.14012. doi:10.1111/cgf.14012.

[35] Marin, R, Rampini, A, Castellani, U, Rodola, E, Ovsjanikov, M, Melzi, S. Instant recovery of shape from spectrum via latent space connections. In: 2020 International Conference on 3D Vision (3DV). IEEE; 2020,URL: https://doi.org/10.1109/3dv50981.2020.00022. doi:10.1109/3dv50981.2020.00022.

[36] Marin, R, Rampini, A, Castellani, U, Rodolà, E, Ovsjanikov, M, Melzi, S. Spectral shape recovery and analysis via data-driven connections. International Journal of Computer Vision 2021;129(10):2745–2760. URL: https://doi.org/10.1007/s11263-021-01492-6. doi:10.1007/s11263-021-01492-6.

[37] Pegoraro, M, Melzi, S, Castellani, U, Marin, R, Rodolà, E. Localized shape modelling with global coherence: An inverse spectral approach. Computer Graphics Forum 2022;41(5):13–24. URL: https://doi.org/10.1111/cgf.14599. doi:10.1111/cgf.14599.

[38] Sharp, N, Attaiki, S, Crane, K, Ovsjanikov, M. DiffusionNet: Discretization agnostic learning on surfaces. ACM Transactions on Graphics 2022;41(3):1–16. URL: https://doi.org/10.1145/3507905. doi:10.1145/3507905.

[39] Gao, L, Lai, YK, Liang, D, Chen, SY, Xia, S. Efficient and flexible deformation representation for data-driven surface modeling. ACM Transactions on Graphics 2016;35(5):1–17. URL: https://doi.org/10.1145/2908736. doi:10.1145/2908736.

[40] Tan, Q, Gao, L, Lai, YK, Xia, S. Variational autoencoders for deforming 3D mesh models 2017;arXiv:1709.04307.

[41] Gao, L, Lai, YK, Yang, J, Zhang, LX, Kobbelt, L, Xia, S. Sparse data driven mesh deformation 2017;arXiv:1709.01250.

[42] Yuan, YJ, Lai, YK, Yang, J, Fu, H, Gao, L. Mesh variational autoencoders with edge contraction pooling. 2019. arXiv:arXiv:1908.02507.

[43] Eisenberger, M, Novotny, D, Kerchenbaum, G, Labatut, P, Neverova, N, Cremers, D, et al. NeuroMorph: Unsupervised shape interpolation and correspondence in one go. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2021,URL: https://doi.org/10.1109/cvpr46437.2021.00739. doi:10.1109/cvpr46437.2021.00739.

[44] Hartman, E, Pierson, E, Bauer, M, Charon, N, Daoudi, M. Bare-esa: A riemannian framework for unregistered human body shapes. ArXiv 2022;abs/2211.13185.

[45] Wang, J, Hertzmann, A, Fleet, DJ. Gaussian process dynamical models. In: Weiss, Y, Schölkopf, B, Platt, J, editors. Advances in Neural Information Processing Systems; vol. 18. MIT Press; 2005,URL: https://proceedings.neurips.cc/paper_files/paper/2005/file/ccd45007df44dd0f12098f486e7e8a0f-Paper.pdf.

[46] Taylor, GW, Hinton, GE, Roweis, ST. Modeling human motion using binary latent variables. In: NIPS. 2006,.

[47] Lehrmann, AM, Gehler, PV, Nowozin, S. Efficient nonlinear markov models for human motion. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. IEEE; 2014,URL: https://doi.org/10.1109/cvpr.2014.171. doi:10.1109/cvpr.2014.171.

[48] Fragkiadaki, K, Levine, S, Felsen, P, Malik, J. Recurrent network models for human dynamics. In: 2015 IEEE International Conference on Computer Vision (ICCV). IEEE; 2015,URL: https://doi.org/10.1109/iccv.2015.494. doi:10.1109/iccv.2015.494.

[49] Jain, A, Zamir, AR, Savarese, S, Saxena, A. Structural-RNN: Deep learning on spatio-temporal graphs. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2016,URL: https://doi.org/10.1109/cvpr.2016.573. doi:10.1109/cvpr.2016.573.

[50] Martinez, J, Black, MJ, Romero, J. On human motion prediction using recurrent neural networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2017,URL: https://doi.org/10.1109/cvpr.2017.497. doi:10.1109/cvpr.2017.497.

[51] Chiu, HK, Adeli, E, Wang, B, Huang, DA, Niebles, JC. Action-agnostic human pose forecasting. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV). IEEE; 2019,URL: https://doi.org/10.1109/wacv.2019.00156. doi:10.1109/wacv.2019.00156.

[52] Liu, Z, Wu, S, Jin, S, Liu, Q, Lu, S, Zimmermann, R, et al. Towards natural and accurate future motion prediction of humans and animals. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2019,URL: https://doi.org/10.1109/cvpr.2019.01024. doi:10.1109/cvpr.2019.01024.

[53] Butepage, J, Black, MJ, Kragic, D, Kjellstrom, H. Deep representation learning for human motion prediction and classification. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2017,URL: https://doi.org/10.1109/cvpr.2017.173. doi:10.1109/cvpr.2017.173.

[54] Butepage, J, Kjellstrom, H, Kragic, D. Anticipating many futures: Online human motion prediction and generation for human-robot interaction. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE; 2018,URL: https://doi.org/10.1109/icra.2018.8460651. doi:10.1109/icra.2018.8460651.

[55] Hernandez, A, Gall, J, Moreno, F. Human motion prediction via spatio-temporal inpainting. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE; 2019,URL: https://doi.org/10.1109/iccv.2019.00723. doi:10.1109/iccv.2019.00723.

[56] Gui, LY, Wang, YX, Liang, X, Moura, JMF. Adversarial geometry-aware human motion prediction. In: Computer Vision – ECCV 2018. Springer International Publishing; 2018, p. 823–842. URL: https://doi.org/10.1007/978-3-030-01225-0_48. doi:10.1007/978-3-030-01225-0_48.

[57] Lebailly, T, Kiciroglu, S, Salzmann, M, Fua, P, Wang, W. Motion prediction using temporal inception module. In: Computer Vision – ACCV 2020. Springer International Publishing; 2021, p. 651–665. URL: https://doi.org/10.1007/978-3-030-69532-3_39. doi:10.1007/978-3-030-69532-3_39.

[58] Ma, T, Nie, Y, Long, C, Zhang, Q, Li, G. Progressively generating better initial guesses towards next stages for high-quality human motion prediction. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2022,URL: https://doi.org/10.1109/cvpr52688.2022.00633. doi:10.1109/cvpr52688.2022.00633.

[59] Vaswani, A, Shazeer, N, Parmar, N, Uszkoreit, J, Jones, L, Gomez, AN, et al. Attention is all you need. In: Advances in Neural Information Processing Systems. 2017, p. 5998–6008.

[60] Tang, Y, Ma, L, Liu, W, Zheng, WS. Long-term human motion prediction by modeling motion context and enhancing motion dynamic. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence. IJCAI'18; AAAI Press. ISBN 9780999241127; 2018, p. 935–941.

[61] Aksan, E, Kaufmann, M, Cao, P, Hilliges, O. A spatio-temporal transformer for 3d human motion prediction. In: 2021 International Conference on 3D Vision (3DV). IEEE; 2021,URL: https://doi.org/10.1109/3dv53792.2021.00066. doi:10.1109/3dv53792.2021.00066.

[62] Cai, Y, Huang, L, Wang, Y, Cham, TJ, Cai, J, Yuan, J, et al. Learning progressive joint propagation for human motion prediction. In: Computer Vision – ECCV 2020. Springer International Publishing; 2020, p. 226–242. URL: https://doi.org/10.1007/978-3-030-58571-6_14. doi:10.1007/978-3-030-58571-6_14.

[63] Duan, Y, Shi, T, Zou, Z, Lin, Y, Qian, Z, Zhang, B, et al. Single-shot

motion completion with transformer. ArXiv 2021;abs/2103.00776.

[64] Petrovich, M, Black, MJ, Varol, G. TEMOS: Generating diverse human motions from textual descriptions. In: Lecture Notes in Computer Science. Springer Nature Switzerland; 2022, p. 480–497. URL: https://doi.org/10.1007/978-3-031-20047-2_28. doi:10.1007/978-3-031-20047-2_28.

[65] Tevet, G, Raab, S, Gordon, B, Shafir, Y, Cohen-Or, D, Bermano, AH. Human motion diffusion model. ArXiv 2022;abs/2209.14916.

[66] Sohn, K, Lee, H, Yan, X. Learning structured output representation using deep conditional generative models. In: Advances in neural information processing systems. 2015, p. 3483–3491.

[67] Zuffi, S, Kanazawa, A, Jacobs, D, Black, MJ. 3D menagerie: Modeling the 3D shape and pose of animals. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). 2017,.

[68] Abrevaya, VF, Manandhar, S, Hétroy-Wheeler, F, Wuhrer, S. A 3dt laplace operator for temporal mesh sequences. Computers &amp Graphics 2016;58:12–22. URL: https://doi.org/10.1016/j.cag.2016.05.018. doi:10.1016/j.cag.2016.05.018.

[69] Mahmood, N, Ghorbani, N, Troje, NF, Pons-Moll, G, Black, MJ. AMASS: Archive of motion capture as surface shapes. In: International Conference on Computer Vision. 2019, p. 5442–5451.

[70] Ionescu, C, Papava, D, Olaru, V, Sminchisescu, C. Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments. IEEE Trans Pattern Anal Mach Intell 2014;36(7):1325–1339.

[71] Welford, BP. Note on a method for calculating corrected sums of squares and products. Technometrics 1962;4(3):419–420. URL: https://doi.org/10.1080/00401706.1962.10490022. doi:10.1080/00401706.1962.10490022.

[72] Community, BO. Blender - a 3D modelling and rendering package. Blender Foundation; Stichting Blender Foundation, Amsterdam; 2018. URL: http://www.blender.org.

[73] Ovsjanikov, M, Ben-Chen, M, Solomon, J, Butscher, A, Guibas, L. Functional maps. ACM Transactions on Graphics 2012;31(4):1–11. URL: https://doi.org/10.1145/2185520.2185526. doi:10.1145/2185520.2185526.

[74] Pavlakos, G, Choutas, V, Ghorbani, N, Bolkart, T, Osman, AAA, Tzionas, D, et al. Expressive body capture: 3D hands, face, and body from a single image. In: Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). 2019, p. 10975–10985.